高职高专计算机教学改革新体系规划教材

Android项目开发教程

主编：车金庆 何征天

清华大学出版社

高职高专计算机教学改革新体系规划教材

Android项目开发教程

车金庆　何征天　　　主　编
李　琳　严正宇　周凌翱　副主编

清华大学出版社
北　京

内 容 简 介

本书以 Android 开发环境为核心，以多个业务相对独立但知识彼此关联的项目应用开发和实现为主线，以 Android 开发环境中各个核心功能的实现为主体内容，以项目实战结合工作任务分解的方式组织内容，完成项目化教学。每个项目应用开发都包括项目分析、算法流程设计、界面设计、代码编写、系统运行与效果测试六个关键环节的内容，将具体的 Android 项目开发与程序设计工程师的岗位工作过程相融合，让读者在实践中能够从技术和职业两种不同的视角掌握 Android 项目开发的全过程。

本书内容的顺序和层次按照 Android 开发环境的难易程度及 Android 应用的复杂程度来编排，共分为 9 章，介绍如何构建 Android 开发环境、实现通信功能、实现图像与动画功能、网络聊天功能、短信管理功能、影音播放功能、地图 GPS 功能等项目任务，并在任务实施过程中全程引入平行的项目实践内容，以供学习者参考与实践。

本书适合作为高等职业院校软件技术专业及相关专业师生的教学参考用书，同时也可以作为移动程序开发爱好者及企业移动应用维护人员的指导用书。

图书在版编目(CIP)数据

Android 项目开发教程/车金庆，何征天主编. —北京：清华大学出版社，2017
(高职高专计算机教学改革新体系规划教材)
ISBN 978-7-302-45555-4

Ⅰ. ①A… Ⅱ. ①车… ②何… Ⅲ. ①移动终端－应用程序－程序设计－高等学校－教材
Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字(2016)第 277468 号

责任编辑：吴梦佳
封面设计：田晓媛
责任校对：袁　芳
责任印制：沈　露

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62770175-4278
印 装 者：北京国马印刷厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：17.75　　**字　　数**：406 千字
版　　次：2017 年 2 月第 1 版　　**印　　次**：2017 年 2 月第1 次印刷
印　　数：1～2000
定　　价：35.00 元

产品编号：063565-01

前言

FOREWORD

本书应用现代职业教育的最新理念，注重培养学生的创新意识、逻辑思维能力和程序代码编写的实践能力，力图呈现新颖的现代职业教育课程观、学习观、教学观、传播观和教材观，同时较好地贯彻落实了以全面素质教育为基础、能力为本位的教学指导思想，是一本面向职业院校的教材。

Android是谷歌和开放手机联盟（Open Handset Alliance，OHA）开发的基于Linux的完整、开放、免费的手机平台。基于Android的应用程序设计易学、易用，极大地降低了在终端设备上开发移动互联应用程序的难度。经过近几年的发展，Android在全球得到了大规模的推广。我国很多高职院校也将“Android开发”作为一门重要的专业课程。

本书以企业技术专家丰富的实践经验为基础、由高职高专专业教师所具备的丰富专业理论知识和实践经验以及教学专家新颖的教学理念为核心进行编写，重点突出了职业能力和综合素质的培养。

本书在内容上依据Android开发环境构建和Android平台基础开发所需要掌握的知识，逐层展开技能点并层层深入，以企业实际开发的Android应用为基础，以程序开发过程的分析、设计、编码和测试为核心主线，同时强调以项目为载体，围绕工作过程重构项目任务，使工作任务的设计和编排既符合企业对程序代码编写人员的实际要求，又能适应高职高专类人才培养的特点，最终实现多个完整的Android应用程序的开发。同时，在修订时，注重引入当前Android程序开发的主流新技术和新理念，使教材内容能够与时俱进。

本书在内容结构设计上突出了工作目标、工作任务及项目实施过程的有机统一，既紧紧围绕核心岗位所需的素质和能力进行阐述，又通过“任务分析”“本章小结”和“项目实践”等加强对学习效果的评估。

本书共9章，按照Android平台的技术体系和项目内容设计难度适宜的工作项目。每个项目又分为若干个工作任务，将相关的理论知识融入项目实践。第1章和第2章主要介绍Android的基础概况及开发环境；第3章是通信功能的设计及开发，实现打电话和发短信的功能，从而使读者掌握基础控件使用、Android布局和应用资源、Android常用控件等知识；第4章是水果连连看的设计及开发，帮助读者学习图像与动画处理；第5章是聊天工具的设计及开发相关控件的使用和聊天功能的实现；第6章是短信智能管理器

的设计及开发，帮助读者学习 Content Provider；第 7 章是学生信息管理系统的设计及开发，帮助读者学习图表；第 8 章是影音播放器的设计及开发，帮助读者学习多媒体；第 9 章是基于百度地图的 GPS 设计及开发，帮助读者学习 GPS 相关知识。

本书由车金庆、何征天编写，李琳、严正宇、周凌翱在编写过程中提出了宝贵的参考意见。

在信息与互联网技术迅速发展之际，编者受水平所限，书中难免存在疏忽和不足之处，敬请读者和同行不吝指正。意见和建议请发电子邮箱 jqche@email.czie.net，谢谢！

编　者

2016 年 9 月

目录

CONTENTS

第 1 章

Android 概述

1.1 智能手机操作系统简介

智能手机操作系统作为智能手机的软件平台，管理智能手机的软硬件资源，为应用软件提供各种必要的服务。每一种智能手机操作系统都有其自身的优点，体系结构以及所能提供的服务也不尽相同，而智能手机本身的特殊性又对智能手机操作系统提出了许多带有共性的需求。比如实时性、开放性、安全性等。现在主流的操作系统主要有 Android（见图 1-1）的 Linux 和苹果的 iOS（见图 1-2），它们的应用软件互不兼容。因为可以像个人计算机一样安装第三方软件，所以智能手机有丰富的功能。智能手机能够显示与个人计算机显示一致的正常网页，具有独立的操作系统以及良好的用户界面，也拥有很强的应用扩展性，能方便随意地安装和删除应用程序。



图 1-1　Android



图 1-2　iOS

1. iOS 简介

iOS 的智能手机操作系统的原名为 iPhone OS，其核心与 Mac OS X 的核心同样都源自 Apple Darwin。iOS 主要是供 iPhone 和 iPod touch 使用。就像其基于的 Mac OS X 操作系统一样，它也是以 Darwin 为基础的。iPhone OS 的系统架构分为四个层次：核心操作系统层（the Core OS layer），核心服务层（the Core Services layer），媒体层（the Media layer），可轻触层（the Cocoa Touch layer）。系统操作占用大概 1.1GB 存储空间。

iOS 由两部分组成：操作系统和能在 iPhone 及 iPod touch 设备上运行原生程序的技术。尽管在底层的实现上 iPhone 与 Mac OS X 共享了一些底层技术，但由于 iPhone 是为

移动终端而开发的，所以要解决的用户需求就与 Mac OS X 有些不同。如果用户是一名 Mac 开发人员，可以在 iPhone OS 中发现很多熟悉的技术，同时也会注意 iPhone OS 的独有之处，比如多触点接口（Multi-Touch interface）和加速器（accelerometer）。

2. Android 简介

Android 是一种以 Linux 为基础的开放源代码操作系统，主要用于便携设备。较多人使用"安卓"。Android 操作系统由 Andy Rubin 开发，最初主要支持手机。2005 年由 Google 收购注资，并组建开放手机联盟开发改良，逐渐扩展到平板电脑及其他领域中。2011 年第一季度，Android 在全球的市场份额跃居第一。

据著名互联网流量监测机构 Net Applications 发布的最新数据显示（见图 1-3），从 2013 年 9 月到 2014 年 7 月，在将近 1 年的时间里，尽管诸如碎片化、安全漏洞等问题让 Android 系统屡遭诟病，但其市场占有率却一直处于稳步攀升状态，从最初的 29.42%升至 44.62%。而 iOS 的使用量却在一路下滑，从 2013 年 9 月的 53.68%降至 44.19%，在与 Android 的比拼中，iOS 首次遭遇了滑铁卢。

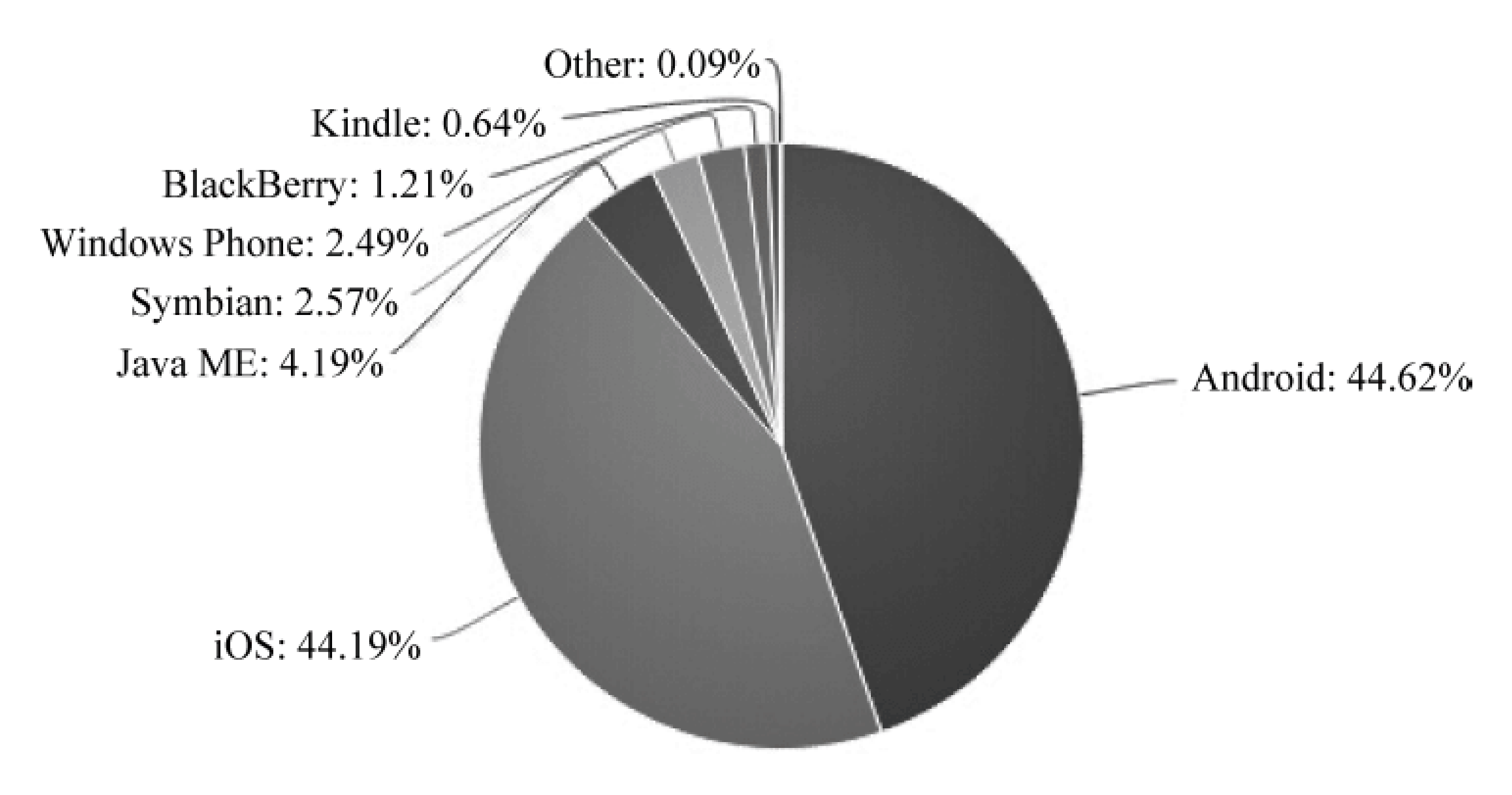


图 1-3　2014 年 7 月各大移动平台分布情况

资料来源：Net Applications

1.2　Android 的基本概念

1.2.1　Android的发展历程

Android 的产生还得从 Andy Rubin（安迪·鲁宾）说起。安迪·鲁宾是硅谷一位著名的程序员，他曾先后在苹果、General Magic、WebTV 等公司工作，2000 年参与创办 Danger 公司。该公司生产的 Hiptop（T-Mobile Sidekick）智能手机具备上网、全键盘和照相功能，2003 年曾在美国风行一时。离开 Danger 之后，安迪·鲁宾创办了新公司，致力于研发手机操作系统。可能是模仿 Linus Torvalds 把自己写的操作系统称为 Linux，安迪·鲁宾的名字是 Andrew，再加上他本身是个机器人迷，所以新公司取名叫作 Android。这就是 Android 的来历。

2005 年 7 月,成立仅 22 个月的 Android 公司被急于开拓无线互联网业务的 Google 收购,安迪·鲁宾也随 Android 加入 Google,继续领导手机操作系统的开发。也就是从这个时候起,业界就开始盛传 Google(谷歌)公司将进军移动通信市场,并推出自主品牌的移动终端产品。更有人将其与苹果公司刚刚推出的 iPhone 相提并论,取名为 GPhone,网络上关于 GPhone 的各种猜想图片也是满天飞。

在沸沸扬扬传了两年多、经过无数次的媒体报道和猜测之后,2007 年 11 月 5 日 Google 终于公布了答案,令人意外的是并没有出现传说中的 Google Phone 或 GPhone。Google 宣布与其他 33 家手机制造商(包括摩托罗拉、宏达电、三星、LG)、手机芯片供货商、软硬件供货商、电信运营商(包括中国移动)联合组成开放手机联盟(Open Handset Alliance),发布了名为 Android 的开放手机软硬件平台。

(1) 里程碑 1:第一款 Android 智能手机。

2008 年 9 月 23 日,Google 与美国电信运营商 T-Mobile 联合,在纽约正式发布第一款 Google 手机——T-Mobile G1。该款手机由宏达电(HTC)制造,内部研发代号为 Dream(中文含义:梦想),是世界上第一部搭载 Android 操作系统的手机。

2009 年 4 月 30 日发布 Android SDK 1.5:Cupcake(纸杯蛋糕)。

(2) 里程碑 2:加入 NDK 支持。

2009 年 9 月 15 日发布 Android SDK 1.6:Donut(甜甜圈)。

2009 年 10 月 26 日发布 Android SDK 2.0。

2010 年 5 月 20 日发布 Android 2.2/2.2.1 Froyo(冻酸奶)。

2010 年 12 月 7 日发布 Android SDK 2.3.x Gingerbread(姜饼)。

(3) 里程碑 3:开始支持平板电脑。

2011 年 2 月 2 日发布 Android SDK 3.0 Honeycomb(蜂巢)。

2011 年 5 月 11 日发布 Android SDK 3.1 Honeycomb(蜂巢)。

2011 年 7 月 13 日发布 Android SDK 3.2 Honeycomb(蜂巢)。

(4) 里程碑 4:手机版和平板电脑版本合并。

2011 年 10 月 19 日在香港发布 Android SDK 4.0 Ice Cream Sandwich(冰激凌三明治)。

2012 年 6 月 28 日发布 Android 4.1 Jelly Bean(果冻豆)。

2012 年 10 月 30 日发布 Android 4.2 Jelly Bean(果冻豆)。

2013 年 9 月 4 日发布 Android 4.4 KitKat(巧克力)。

2014 年 10 月 15 日发布 Android 5.0 Lollipop(棒棒糖)三款新 Nexus 设备——Nexus 6、Nexus 9 平板及 Nexus Player,将率先搭载 Android 5.0,之前的 Nexus 5、Nexus 7 及 Nexus 10 将会很快获得更新。

1.2.2 Android的平台优势

1. 开放性

在优势方面,Android 平台首先就是其开发性,开放的平台允许任何移动终端厂商加

入 Android 联盟。显著的开放性可以使其拥有更多的开发者，随着用户和应用的日益丰富，一个崭新的平台也将很快走向成熟。

开放性对于 Android 的发展而言，有利于积累人气，这里的人气包括消费者和厂商，而对于消费者，最大的受益正是丰富的软件资源。开放的平台也会带来更大的竞争，如此一来，消费者将可以用更低的价位购得心仪的手机。

2. 挣脱运营商的束缚

很长一段时间，特别是在欧美地区，手机应用往往受运营商制约，什么功能接入什么网络，几乎都受运营商的控制。自从 iPhone 上市以来，运营商的制约减少，用户可以更加方便地连接网络。随着 EDGE、HSDPA 这些 2G 至 3G 移动网络的逐步过渡和提升，手机随意接入网络已不是运营商口中的笑谈。而互联网巨头 Google 推动的 Android 终端天生就有网络特色，让用户离互联网更近。

3. 丰富的硬件选择

这一点与 Android 平台的开放性相关，由于 Android 的开放性，众多的厂商会推出千奇百怪、功能特色各具的多种产品。功能上的差异和特色，却不会影响数据同步甚至软件的兼容，如同从诺基亚 Symbian 风格手机一下改用苹果 iPhone，同时还可将 Symbian 中优秀的软件带到 iPhone 使用、联系人等资料更是可以方便地转移。

4. 开发商不受任何限制

Android 平台提供给第三方开发商一个十分宽泛、自由的环境，不会受到各种条条框框的限制，可想而知，会有大量新颖别致的软件诞生。但这也有其两面性，血腥、暴力、情色方面的程序和游戏如何控制是留给 Android 难题之一。

5. Google 应用的无缝结合

互联网的 Google 已经走过 10 年，从搜索巨人到全面的互联网渗透，Google 应用如地图、邮件、搜索等已经成为连接用户和互联网的重要纽带，而 Android 平台手机将无缝结合这些优秀的 Google 应用。

1.3 Android 系统架构

Android 采用了分层架构的思想，如图 1-4 所示。从上层到底层共包括四层，分别是应用程序层(APPLICATIONS)、应用程序框架层(APPLICATION FRAMEWORK)、系统库(LIBRARIES)和 Android 运行时(ANDROID RUNTIME)、Linux 内核(LINUX KERNEL)。

每层功能简要介绍如下。

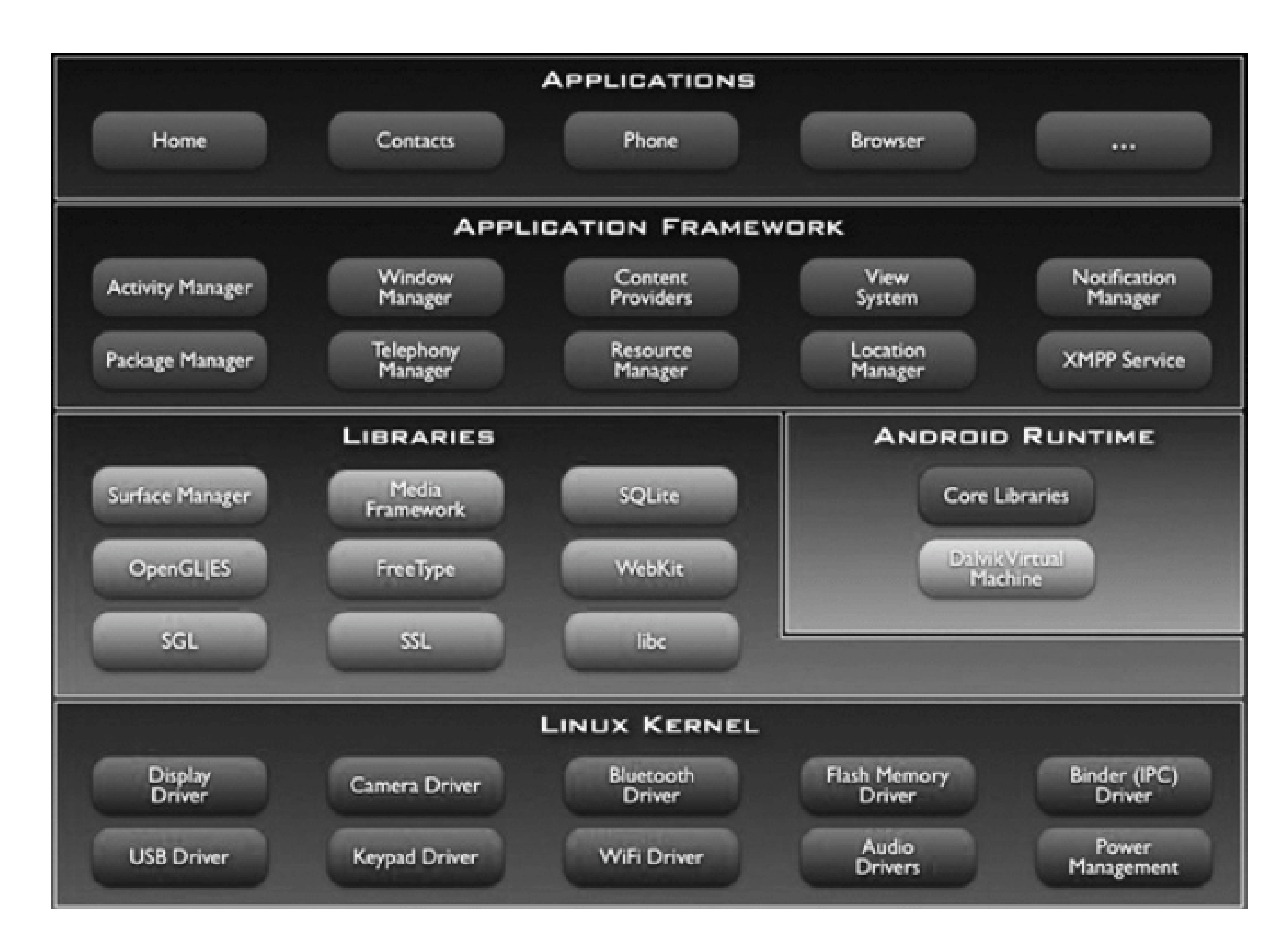


图 1-4 Android 系统架构图

1. 应用程序层

应用程序层提供一些核心应用程序包，例如电子邮件、短信、日历、地图、浏览器和联系人管理等。同时，开发者可以利用 Java 语言设计和编写属于自己的应用程序，而这些程序与那些核心应用程序彼此平等、友好共处。

2. 应用程序框架层

应用程序框架层是 Android 应用开发的基础，包括活动管理器、窗口管理器、内容提供者、视图系统、包管理器、电话管理器、资源管理器、位置管理器、通知管理器和 XMPP 服务十个部分。在 Android 平台上，开发人员可以完全访问核心应用程序所使用的 API 框架。并且任何一个应用程序都可以发布自身的功能模块，而其他应用程序则可以使用这些已发布的功能模块。基于这样的重用机制，用户就可以方便地替换平台本身的各种应用程序组件。

3. 系统库和 Android 运行时(虚拟机)

系统库包括九个子系统，分别是图层管理、媒体库、SQLite、OpenGLEState、FreeType、WebKit、SGL、SSL 和 libc。Android 运行时包括核心库和 Dalvik 虚拟机，前者既兼容了大多数 Java 语言所需要调用的功能函数，又包括了 Android 的核心库，比如 android.os、android.net、android.media 等。后者是一种基于寄存器的 Java 虚拟机，Dalvik 虚拟机主要是完成对生命周期的管理、堆栈的管理、线程的管理、安全和异常的管理以及垃圾回收等重要功能。

4. Linux 内核

Android 核心系统服务依赖于 Linux 2.6 内核，如安全性、内存管理、进程管理、网络

协议栈和驱动模型。Linux 内核也是作为硬件与软件栈的抽象层。驱动包括显示驱动、摄像头驱动、键盘驱动、WiFi 驱动、Audio 驱动、Flash 内存驱动、Binder(IPC)驱动、电源管理等。

5. 总结

(1) Android 的系统架构采用分层架构的思想,架构清晰,层次分明,协同工作。

(2) Android 的系统架构不仅从宏观上介绍了 Android 系统,同时,也给我们的学习与实践指明了方向。若 Android 应用开发,则应该研究 Android 的应用程序框架层和应用程序层;若 Android 系统开发,则应该研究 Android 的系统库和 Android 运行时;若 Android 驱动开发,则应该研究 Android 的 Linux 内核。总之,找准切入点,实践出真知。

1.4 本章小结

本章首先介绍了智能手机操作系统的概念,然后介绍了 Android 的发展轨迹,Android SDK 发布时间点和里程碑,紧接着介绍了 Android 平台优势和系统架构,让大家对 Android 开发有了一定的认识。

第 2 章

Android 开发环境构建

2.1 开发环境搭建

2.1.1 搭建 Android 环境需要安装的软件

1. JDK 8

可以到 http://www.oracle.com/technetwork/java/javase/downloads/index.html 下载，如图 2-1 所示。

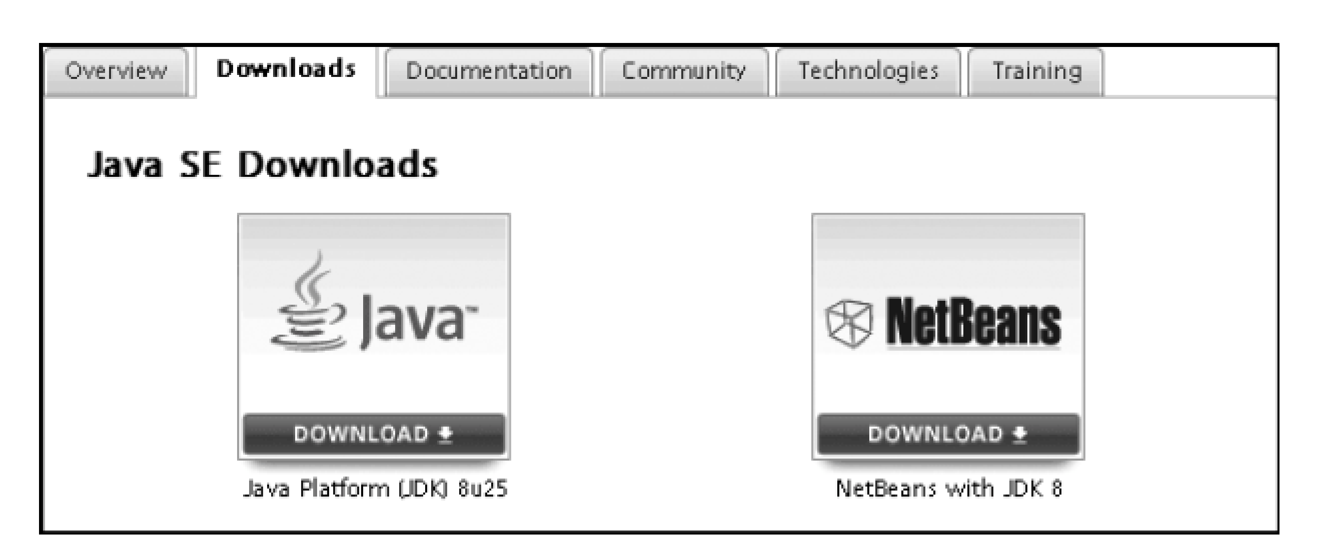
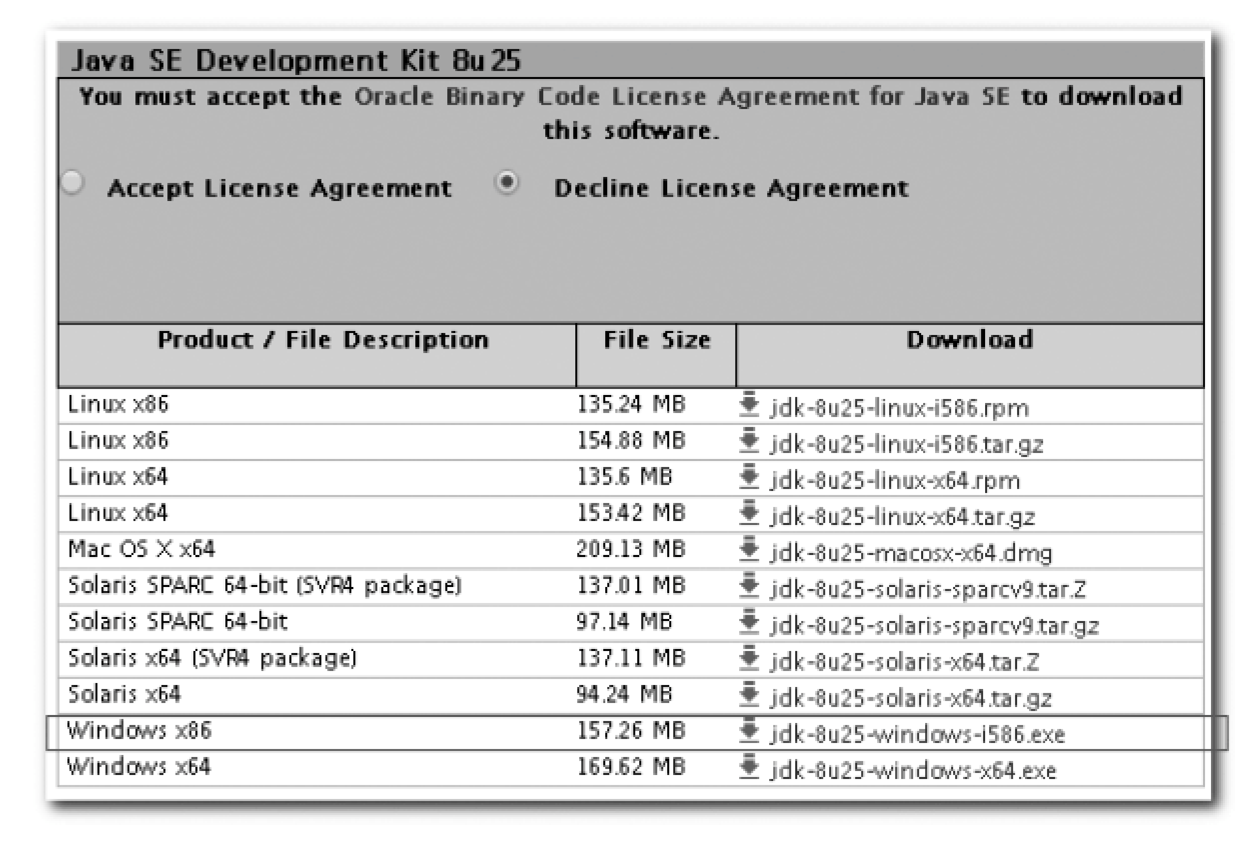


图 2-1 JDK 8 下载页面

单击 Java DOWNLOAD，进入如图 2-2 所示页面。



Java SE Development Kit 8u25

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement　Decline License Agreement

Product / File Description	File Size	Download
Linux x86	135.24 MB	jdk-8u25-linux-i586.rpm
Linux x86	154.88 MB	jdk-8u25-linux-i586.tar.gz
Linux x64	135.6 MB	jdk-8u25-linux-x64.rpm
Linux x64	153.42 MB	jdk-8u25-linux-x64.tar.gz
Mac OS X x64	209.13 MB	jdk-8u25-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	137.01 MB	jdk-8u25-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	97.14 MB	jdk-8u25-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	137.11 MB	jdk-8u25-solaris-x64.tar.Z
Solaris x64	94.24 MB	jdk-8u25-solaris-x64.tar.gz
Windows x86	157.26 MB	jdk-8u25-windows-i586.exe
Windows x64	169.62 MB	jdk-8u25-windows-x64.exe

图 2-2 下载 JDK 8

单击 Accept License Agreement，选择 Windows x86 下载。如果是 64 位机，则选择 Windows x64 进行下载。

2. Eclipse 4.4.1（SR1a）

Eclipse 是一个开源的、基于 Java 的可扩展的集成开发环境（IDE）。Eclipse 中可以集成多种插件，以完成特定语言的开发。

下载地址 http://www.eclipse.org/downloads/页面中的 Eclipse IDE for Eclipse Committers 4.4.1 SR1a，如图 2-3 所示。



图 2-3 Eclipse 下载页面

下载 Windows 32 Bit，如果是 64 位机，下载 Windows 64 Bit，此时进入如图 2-4 所示页面。



图 2-4 下载 Eclipse

单击[China] NEU（http）下载。

3. Android SDK

下载地址：http://dl.google.com/android/android-sdk_r22-windows.zip。

4. ADT Plugin for Eclipse

下载地址：http://dl.google.com/android/ADT-22.0.1.zip。

2.1.2 安装步骤

步骤一：分别解压缩 eclipse-java-luna-SR1-win32.zip 和 android-sdk_r22-windows.zip 到 E:\android-tools \路径下，如图 2-5 所示。

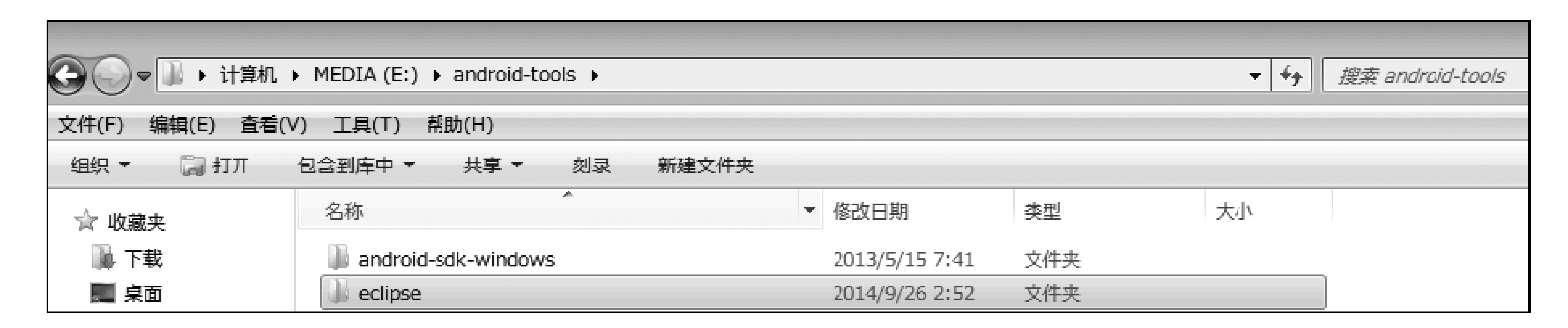


图 2-5 解压缩 eclipse-java-luna-SR1-win32.zip 和 android-sdk_r22-windows.zip

E:\android-tools\eclipse。

E:\android-tools\android-sdk-windows。

步骤二：配置 Android SDK(见图 2-6)。

(1) 执行 E:\android-tools\android-sdk-windows\SDK Manager.exe。

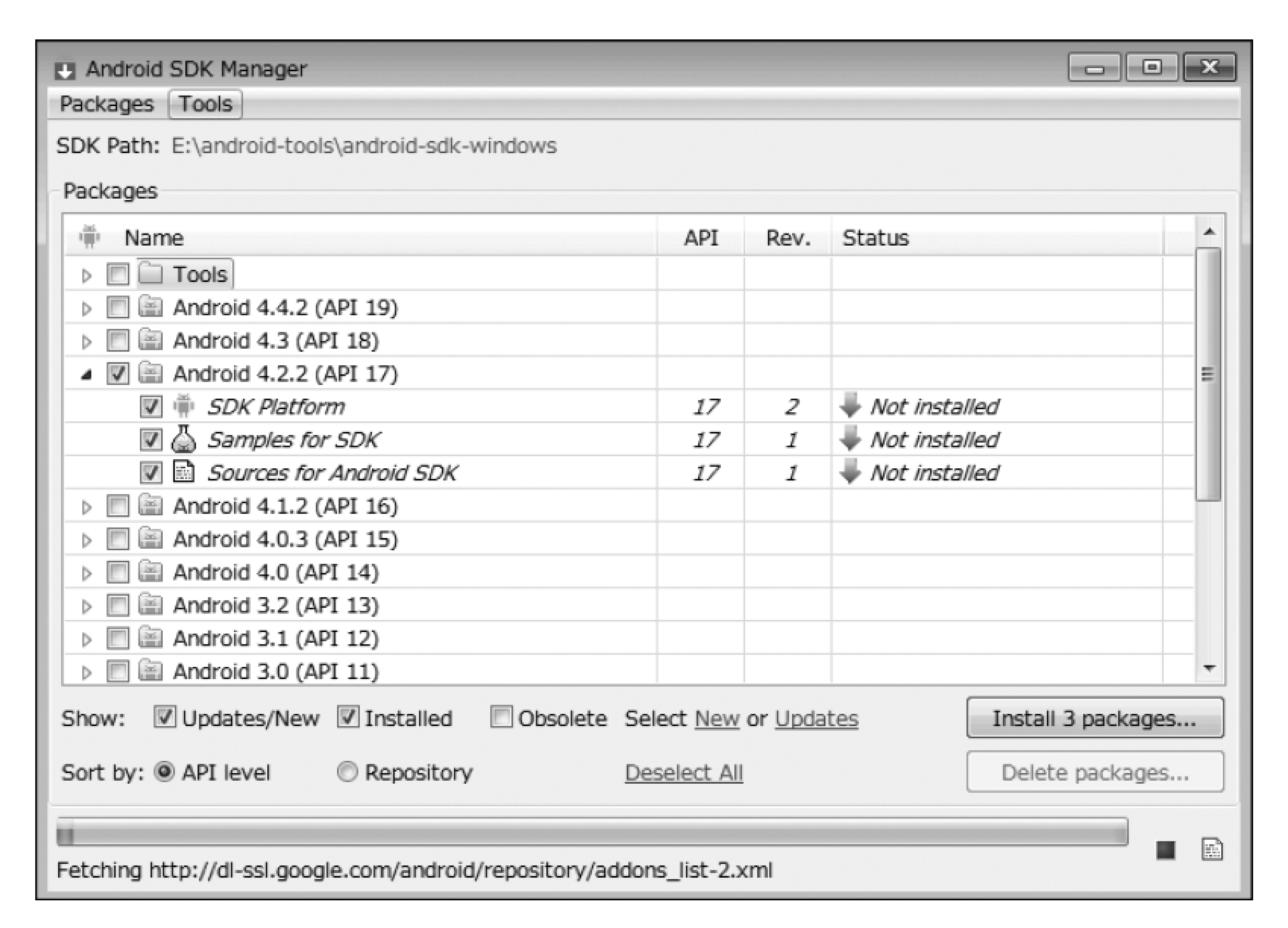


图 2-6 配置 Android SDK

(2) 选择安装包，然后单击 Install 3 packages...。

(3) Accept All 同意全部协议。

(4) 然后单击 Install 开始下载、安装。

步骤三：在 Eclipse 中安装插件 ADT。

(1) 启动 Eclipse，选择菜单 Help→Install New Software，如图 2-7 所示。

(2) 弹出对话框 Install，如图 2-8 所示，选择 Add→Archive...，然后选择本地的 ADT 压缩包文件，在 Name 部分给出命名，本例为 adt4.2。

(3) 在列表中选择 Developer Tools，然后单击 Next 按钮，如图 2-9 和图 2-10 所示。



图 2-7 启动 Eclipse

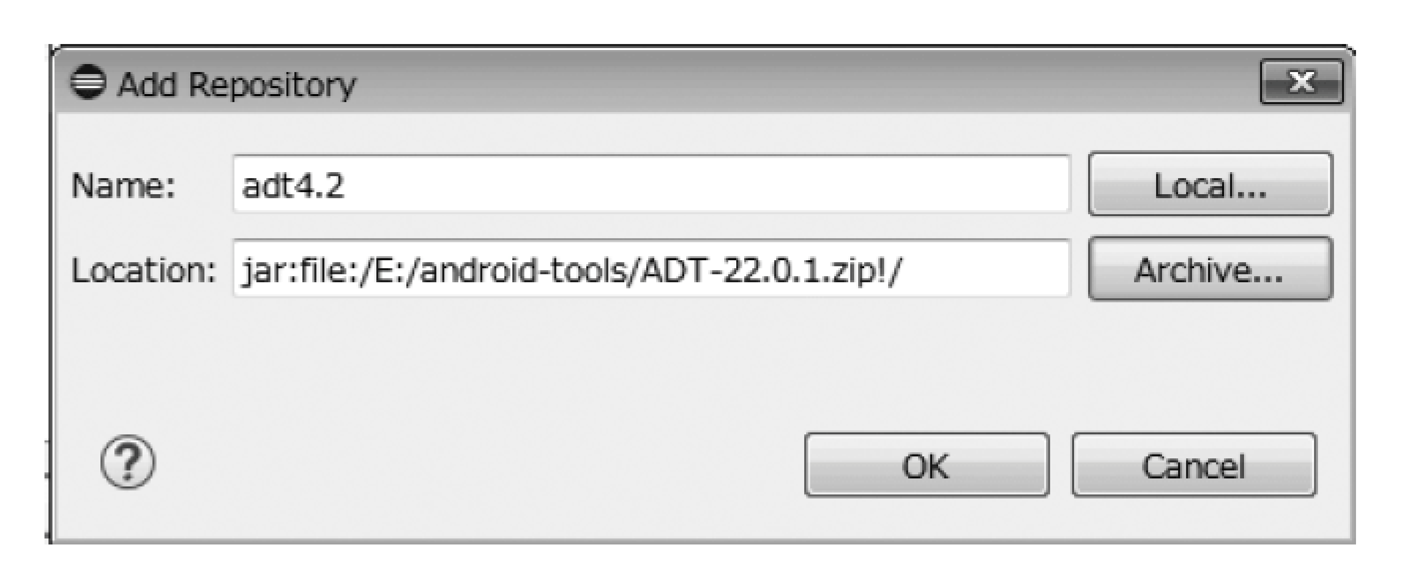


图 2-8 Install 界面

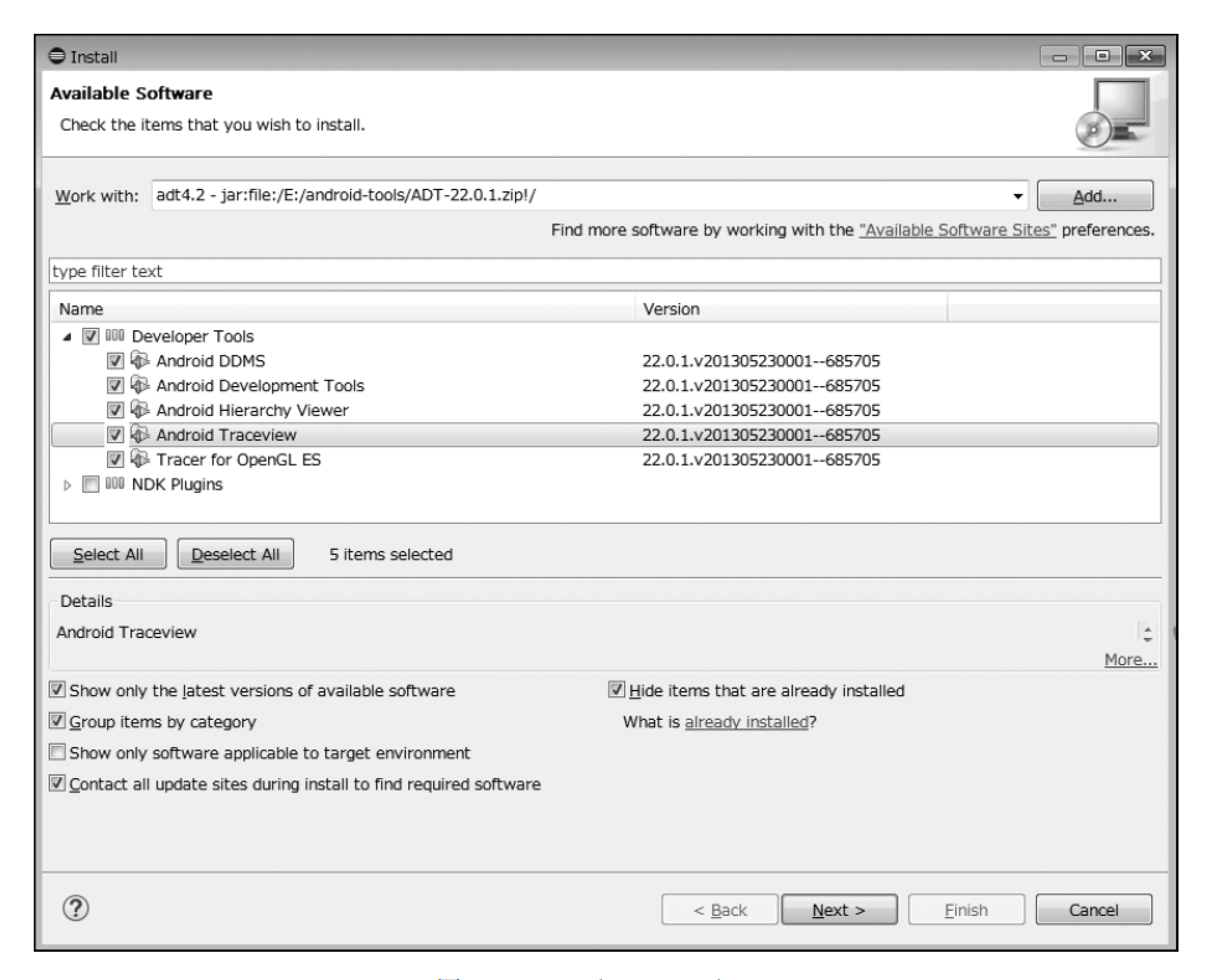


图 2-9 Developer Tools(1)

(4) 接受所有的安装协议,然后单击 Finish 按钮,如图 2-11 和图 2-12 所示。

(5) 安装过程中,出现如图 2-13 所示的警告,单击 OK 按钮。

(6) 安装完毕,提示重新启动 Eclipse,如图 2-14 所示。

2.1.3 Android模拟器运行环境配置

(1) 打开 Eclipse→AVD Manager,如图 2-15 所示。

(2) 新建虚拟机,如图 2-16 所示。

(3) 设置虚拟机默认参数,如图 2-17 所示。

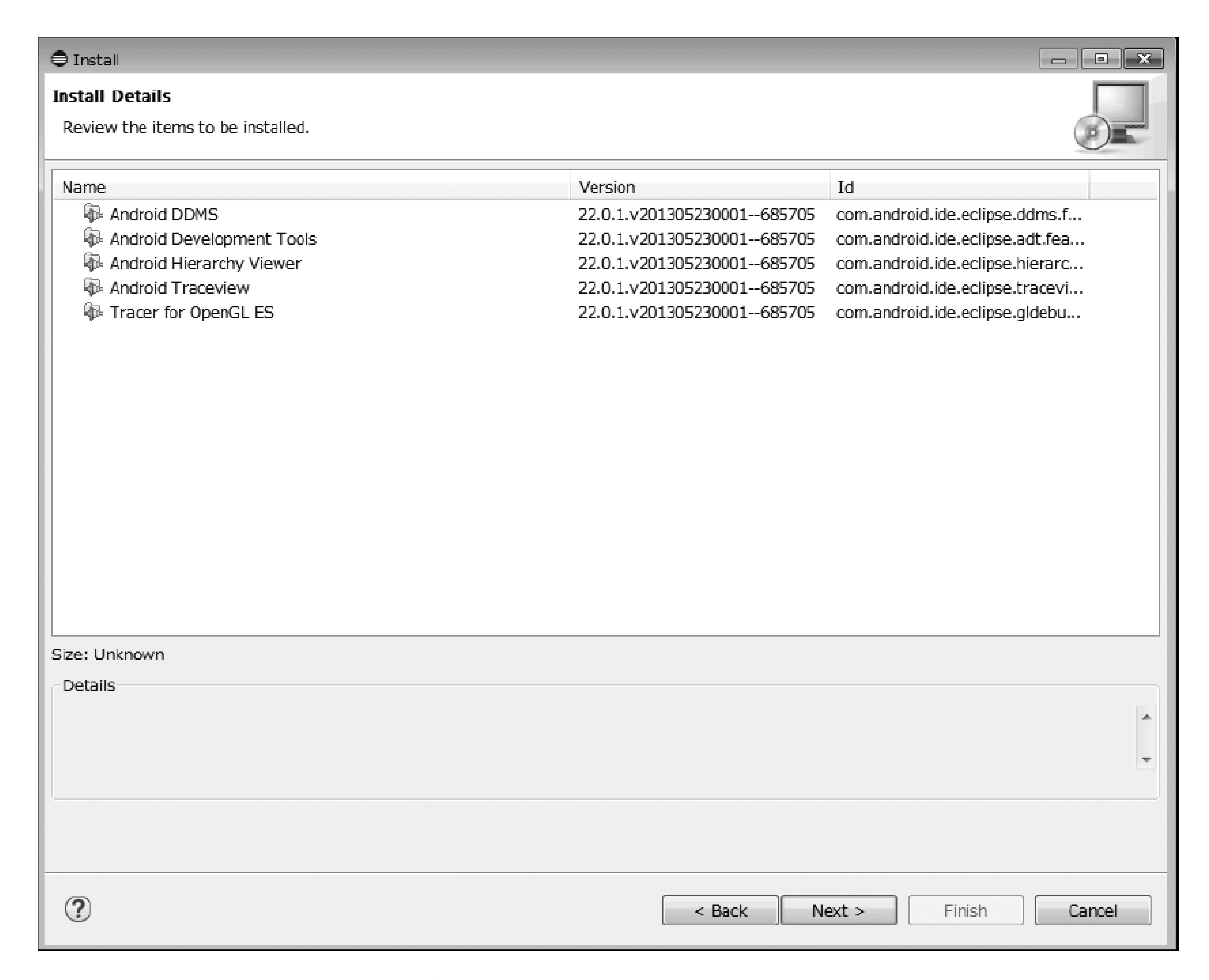


图 2-10 Developer Tools(2)

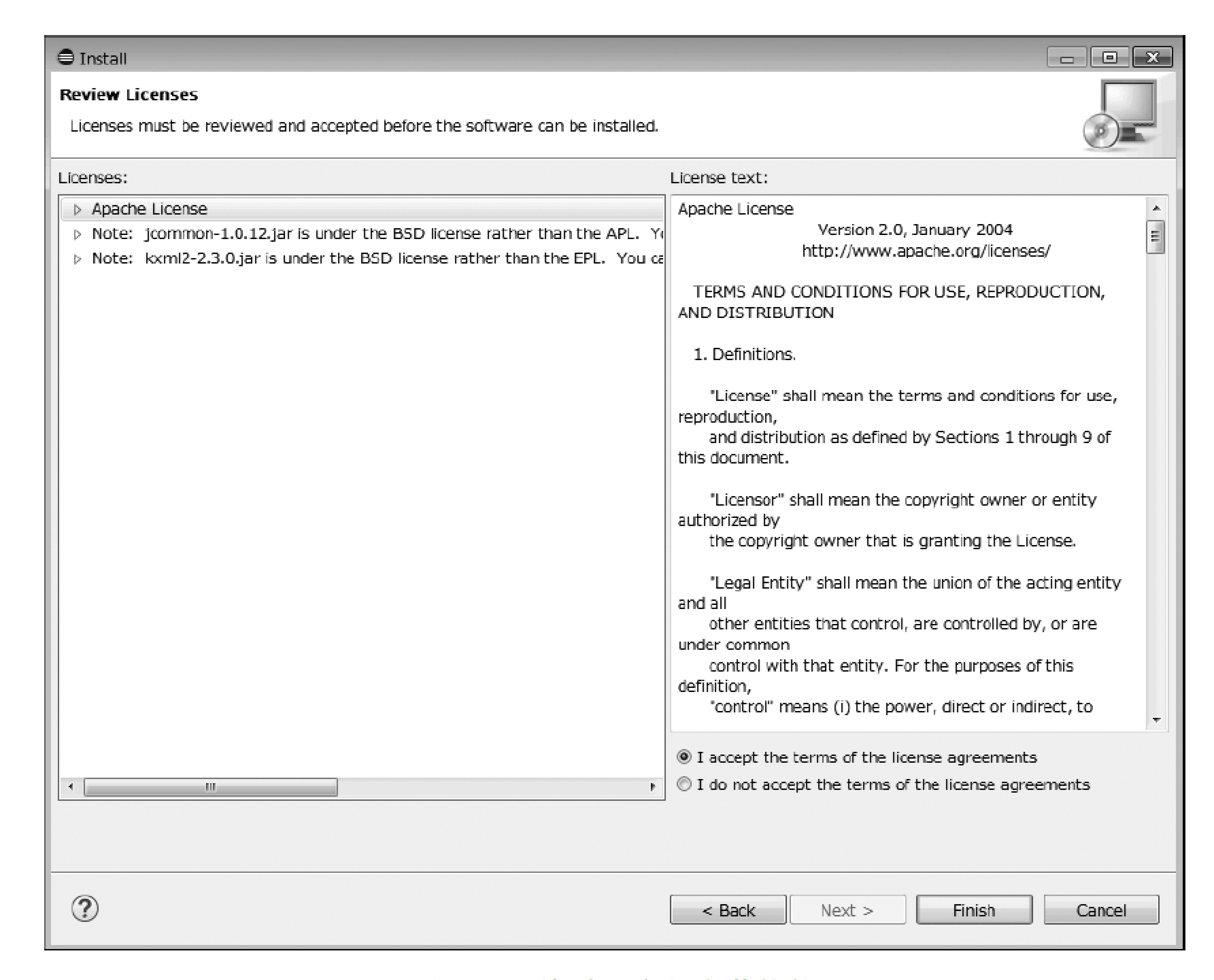


图 2-11 接受所有的安装协议

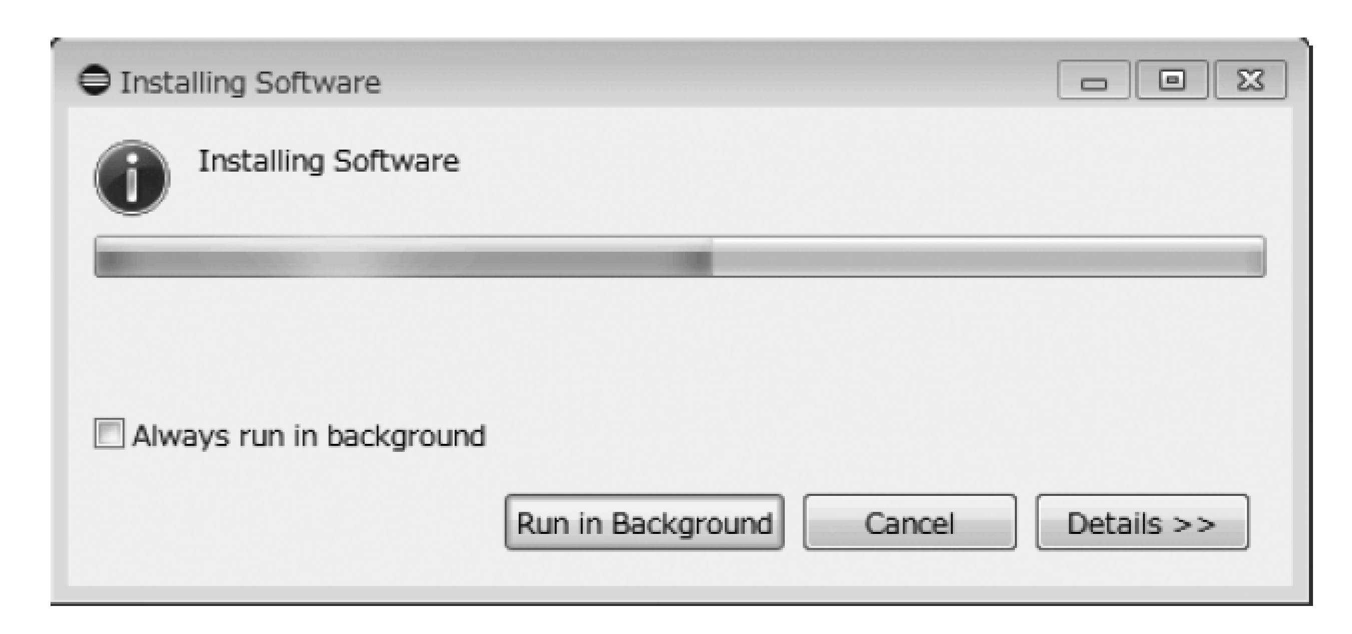


图 2-12 Installing Software

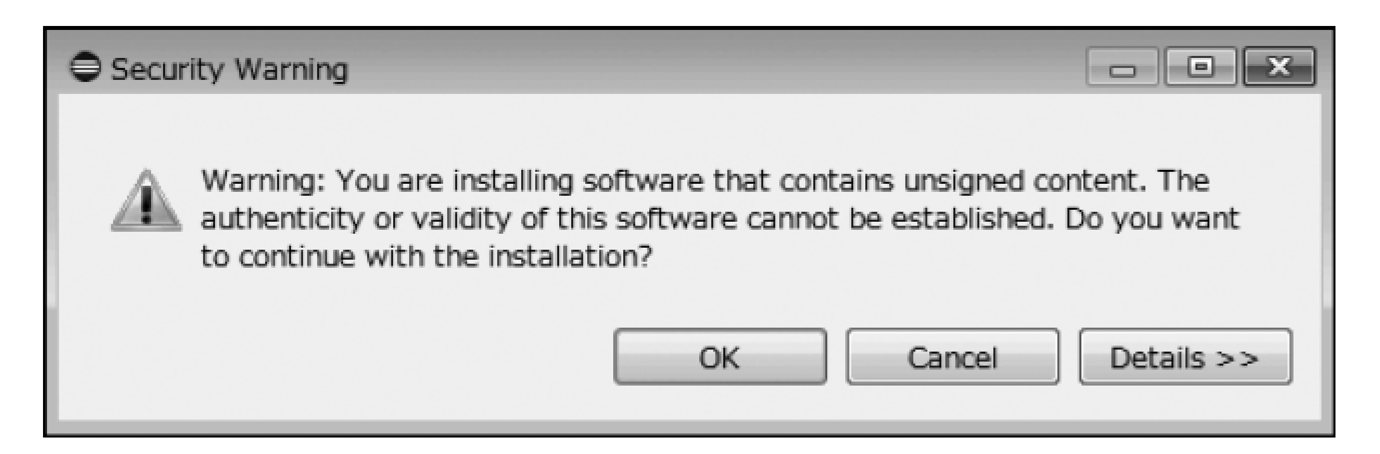


图 2-13 Security Warning

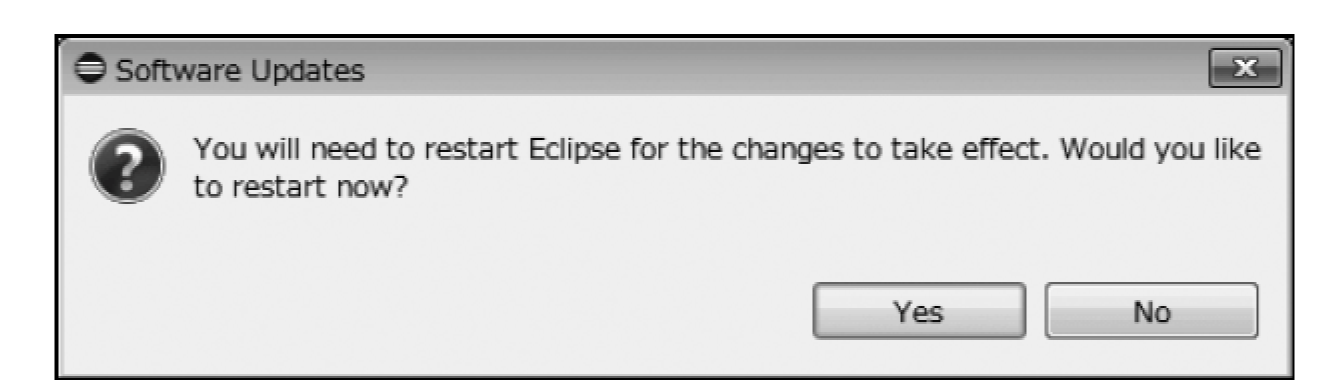


图 2-14 Software Updates

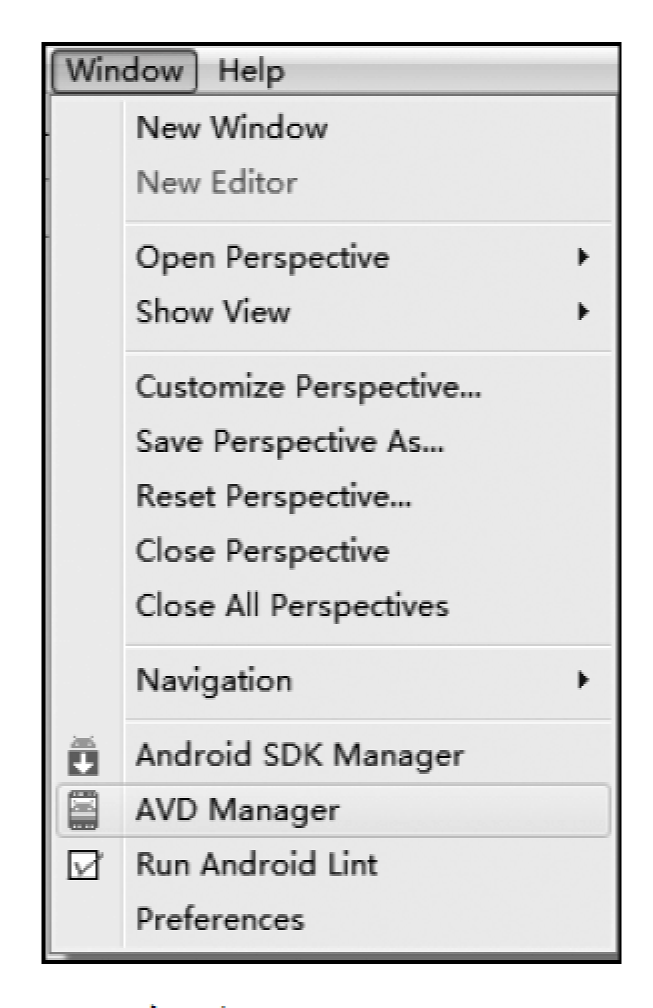


图 2-15 打开 Eclipse→AVD Manager

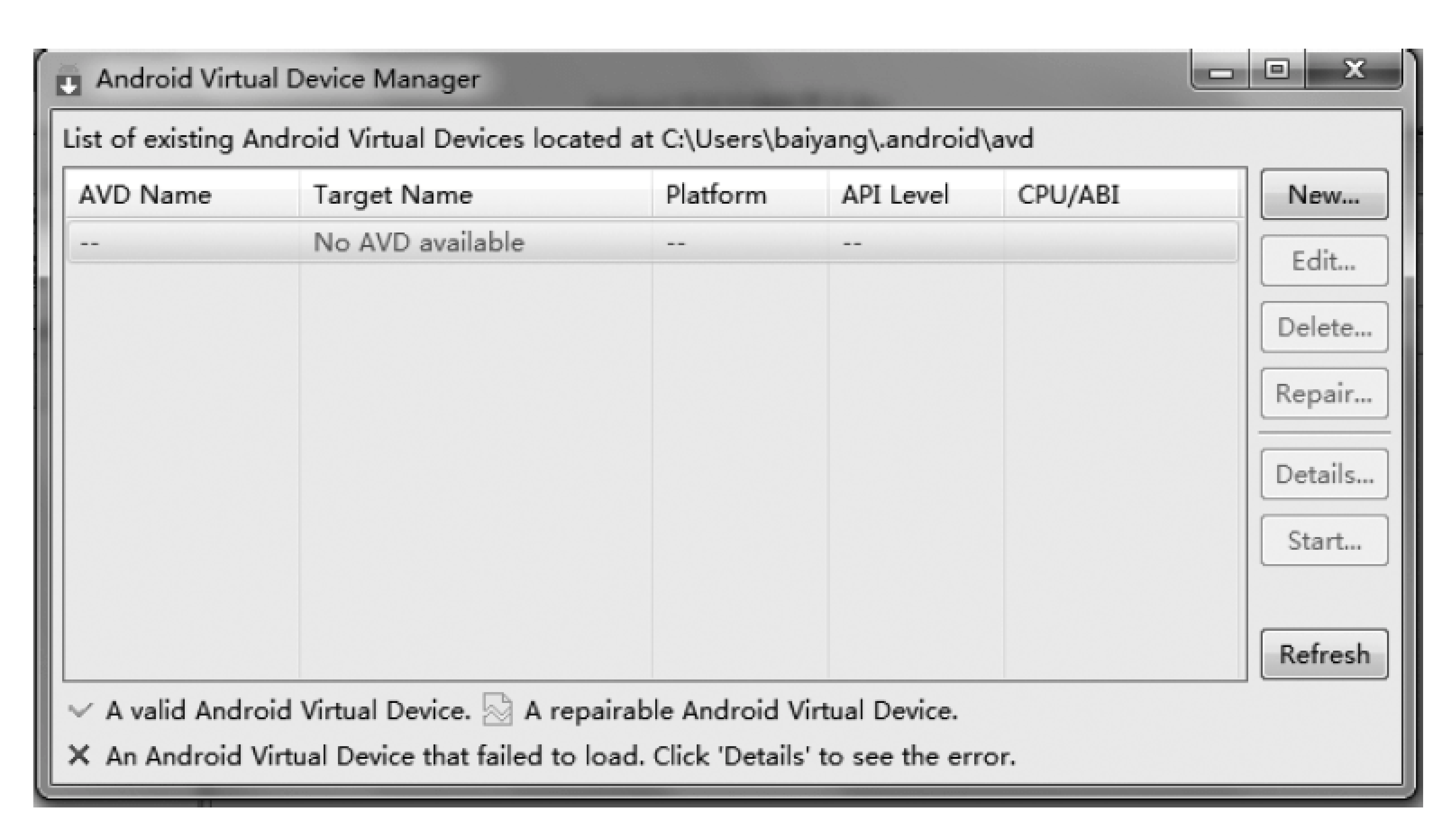


图 2-16　新建虚拟机

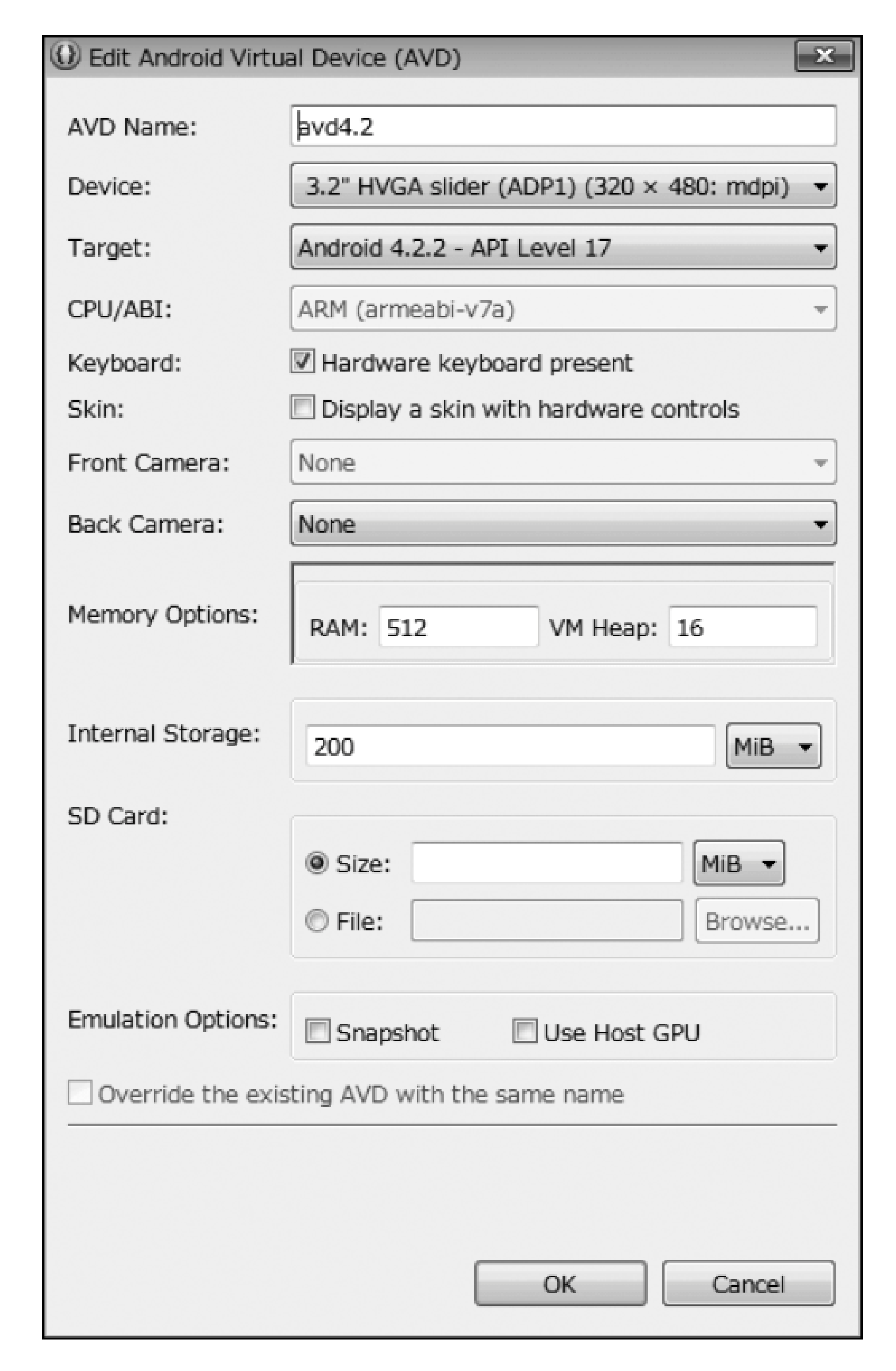


图 2-17　设置虚拟机默认参数

(4) 虚拟机建立完毕，如图 2-18 所示。

(5) 单击 Start 按钮，启动虚拟机，如图 2-19 所示。

(6) Android 虚拟运行环境配置完毕，如图 2-20 所示。

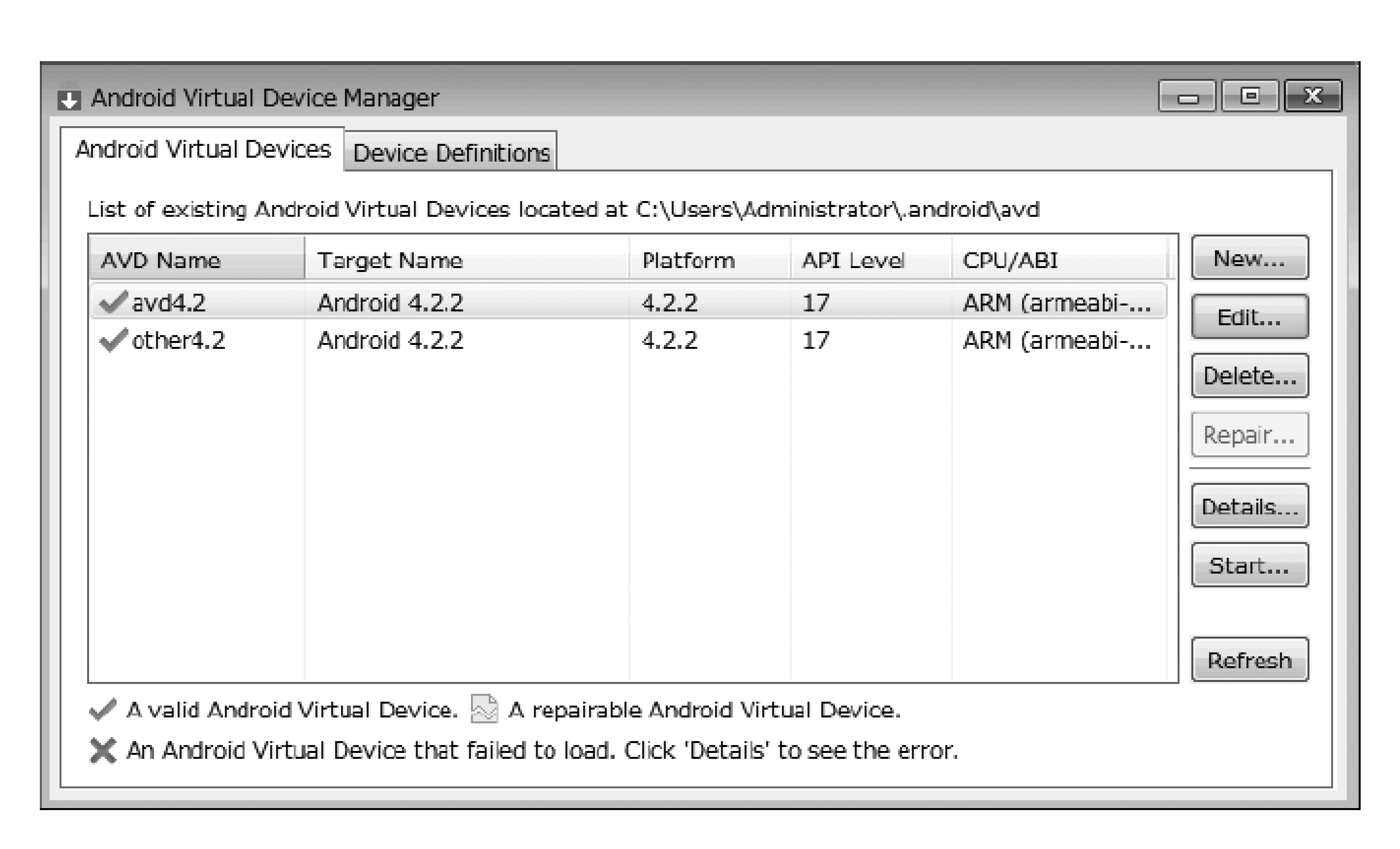


图 2-18 虚拟机建立完毕

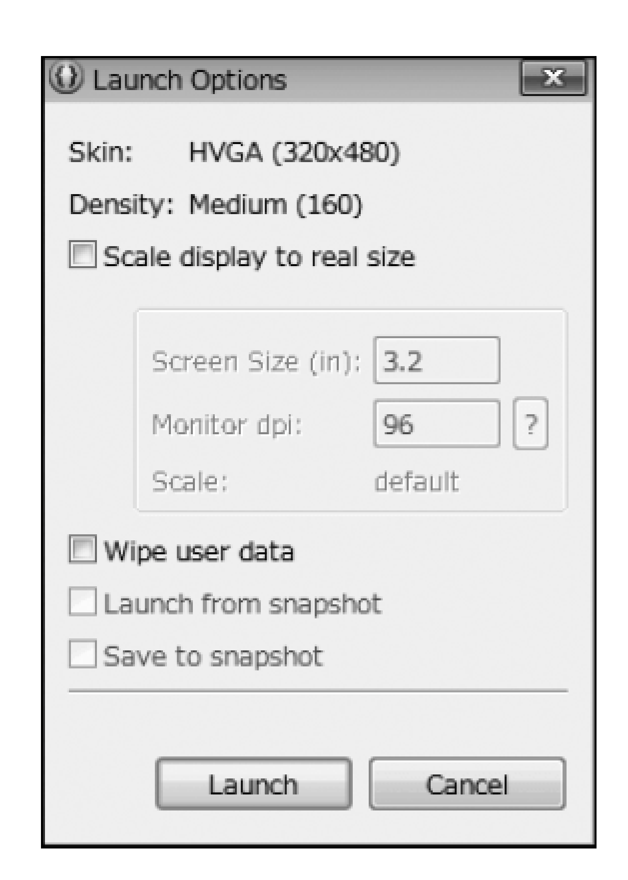


图 2-19 启动虚拟机

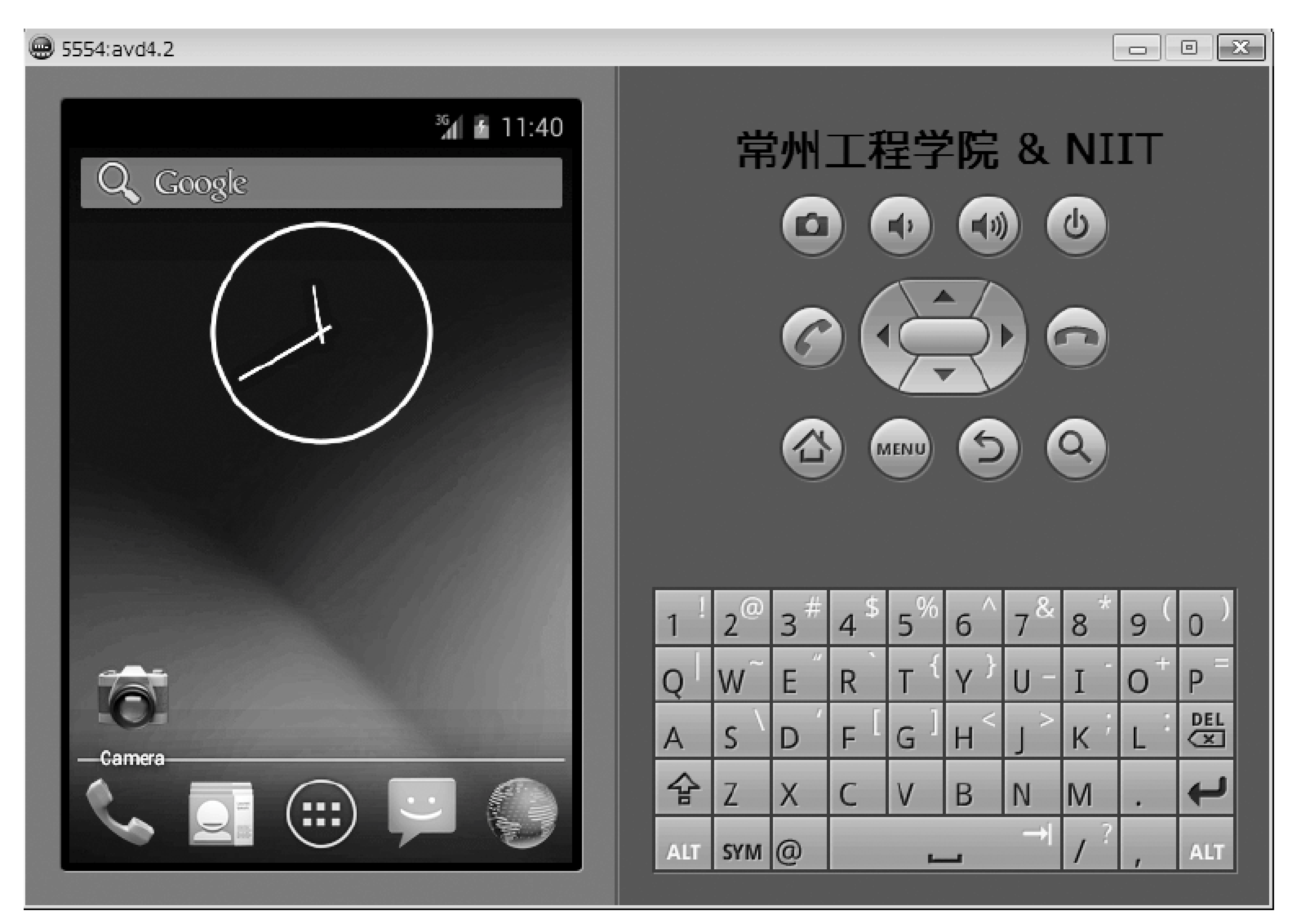


图 2-20 Android 虚拟运行环境配置完毕

2.1.4　新的 Android 开发环境——Android Studio

在 Google 2013 年 I/O 大会上，谷歌推出新的 Android 开发环境——Android Studio（见图 2-21），开发者可以在编写程序的同时看到自己不同尺寸的屏幕。下载地址：http://developer.android.com/sdk/index.html。

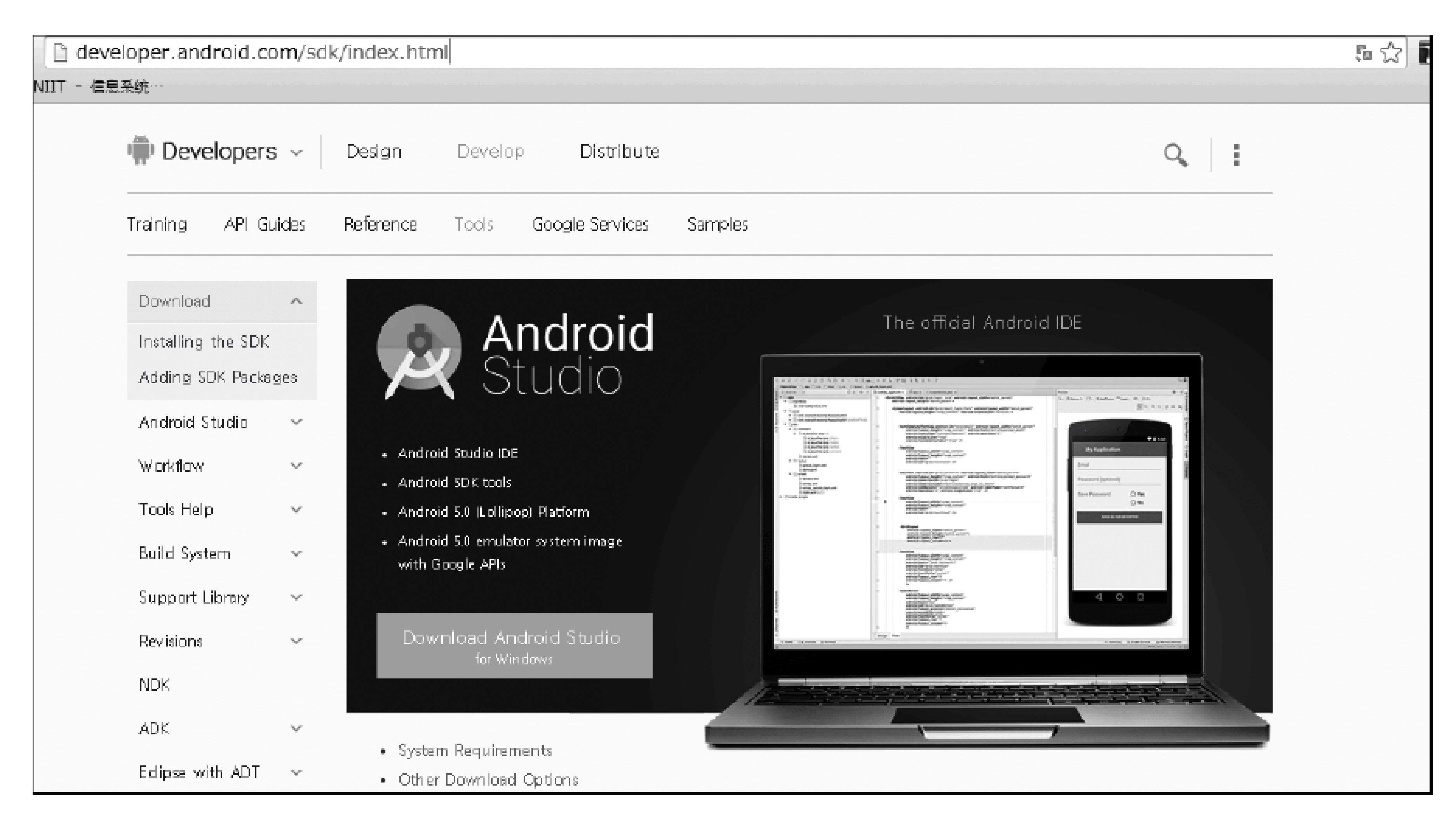


图 2-21　Android Studio

下载后安装，Android Studio 安装需要 JDK 是 1.7 或以上版本。安装过程一直单击“下一步”按钮即可。安装完成如图 2-22 所示。

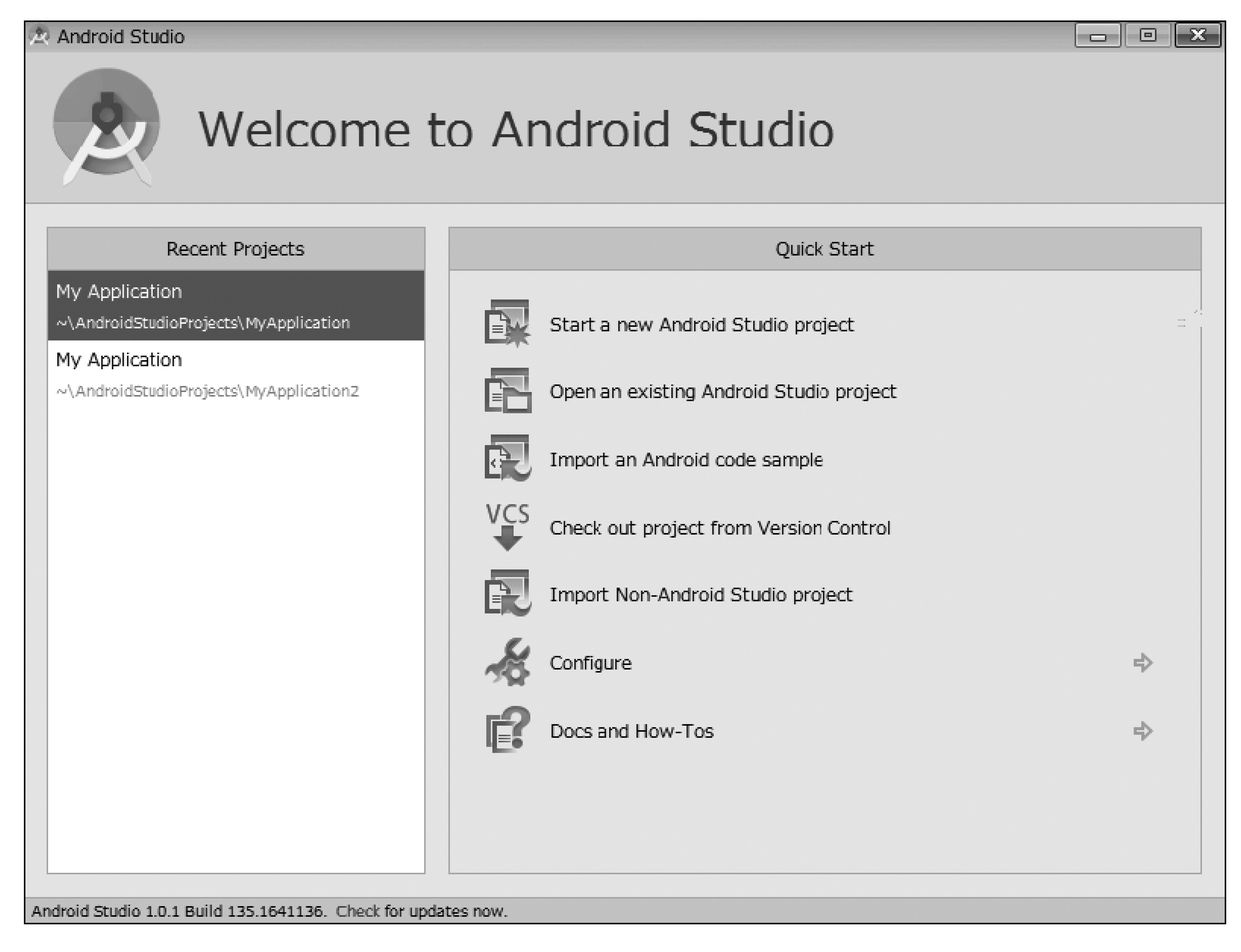


图 2-22　安装完成的 Android Studio

单击 Start a new Android Studio project，新建一个 Android 项目，如图 2-23 所示。

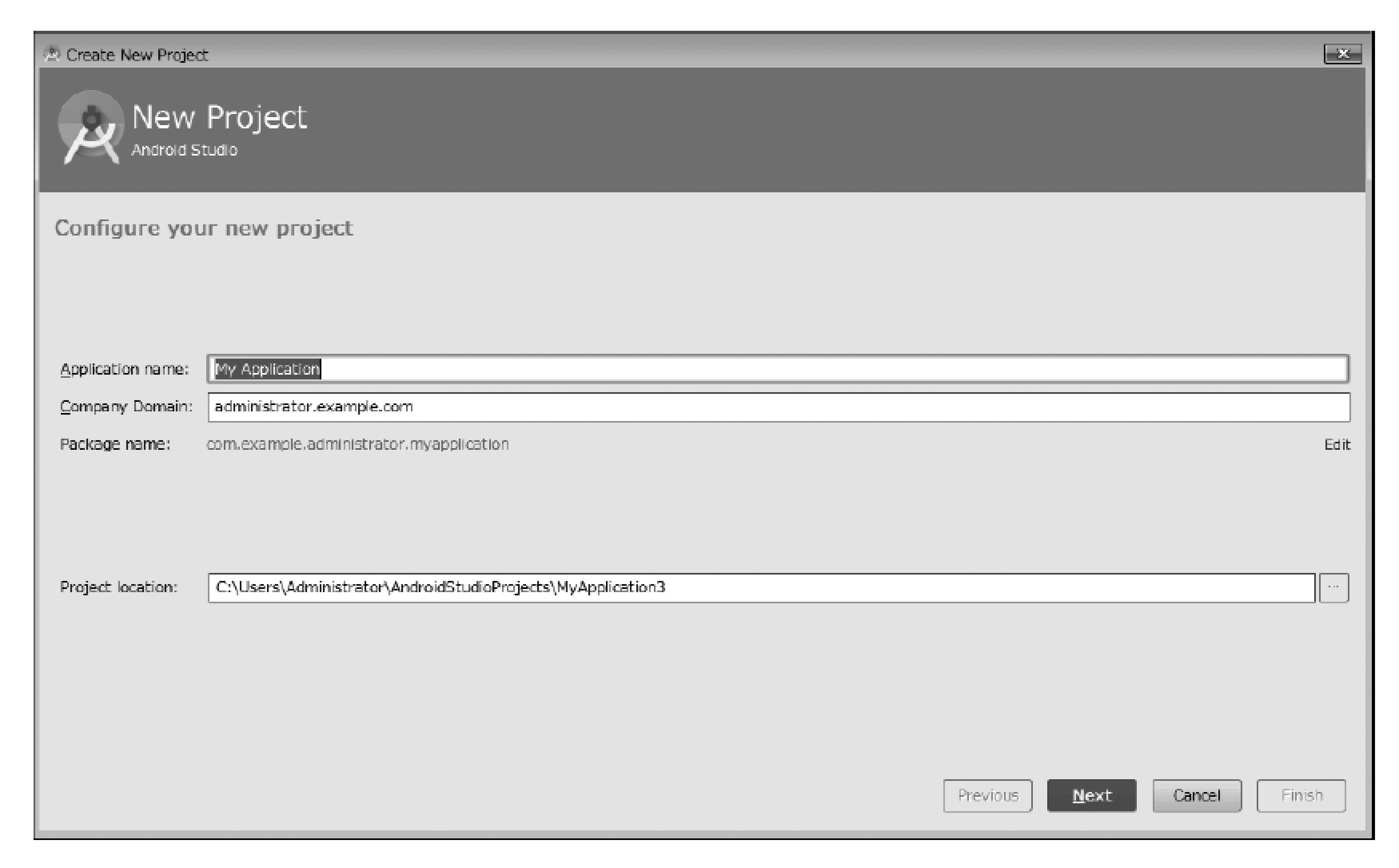


图 2-23　新建一个 Android 项目

一直单击 Next 按钮，直到单击 Finish 按钮结束，项目创建完成，如图 2-24 所示。

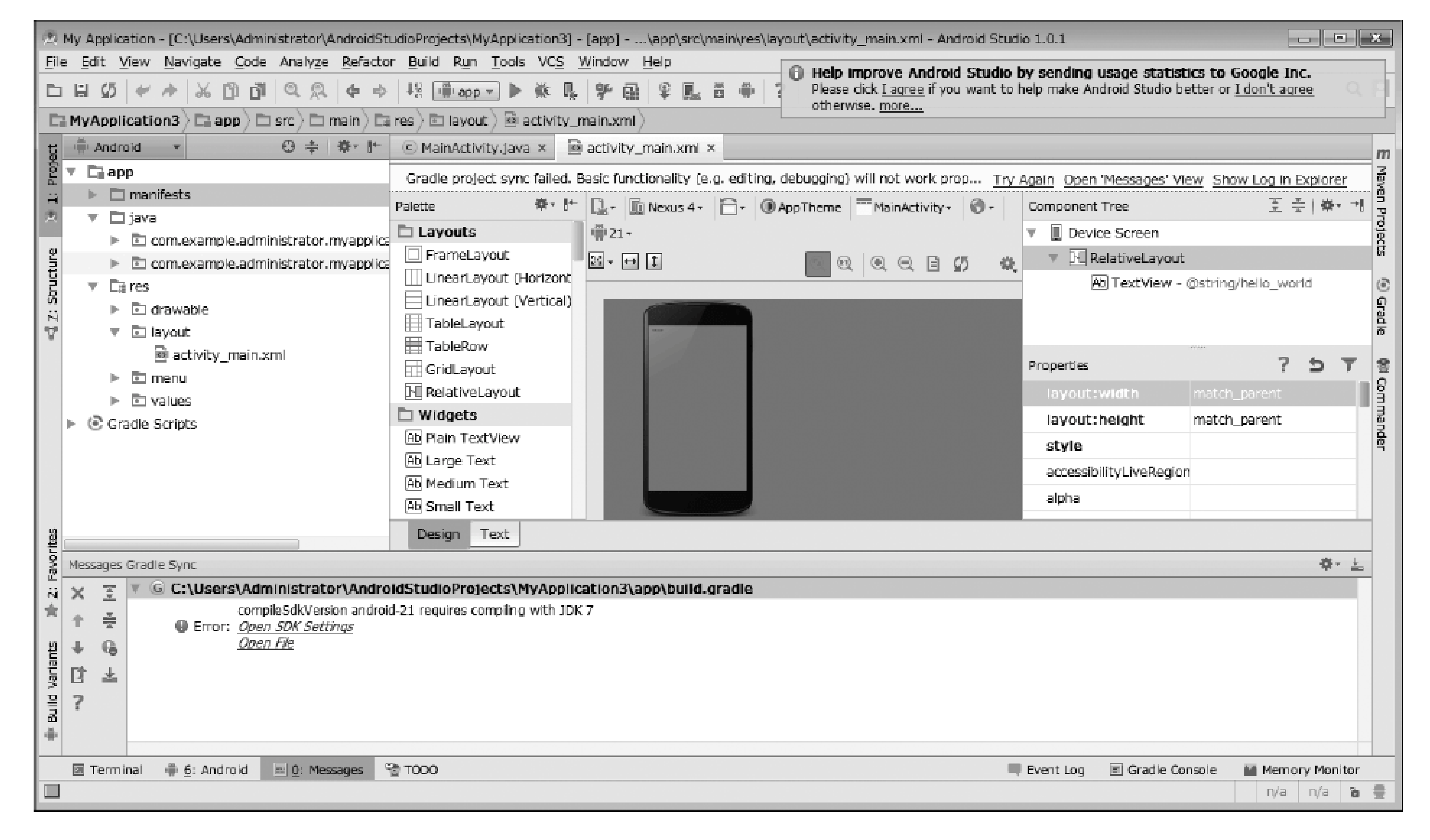


图 2-24　项目创建完成

2.2　创建 Android 应用程序

Android 第一个程序：Hello Android。

（1）打开 Eclipse→File→Project→New Android Application，如图 2-25 所示。

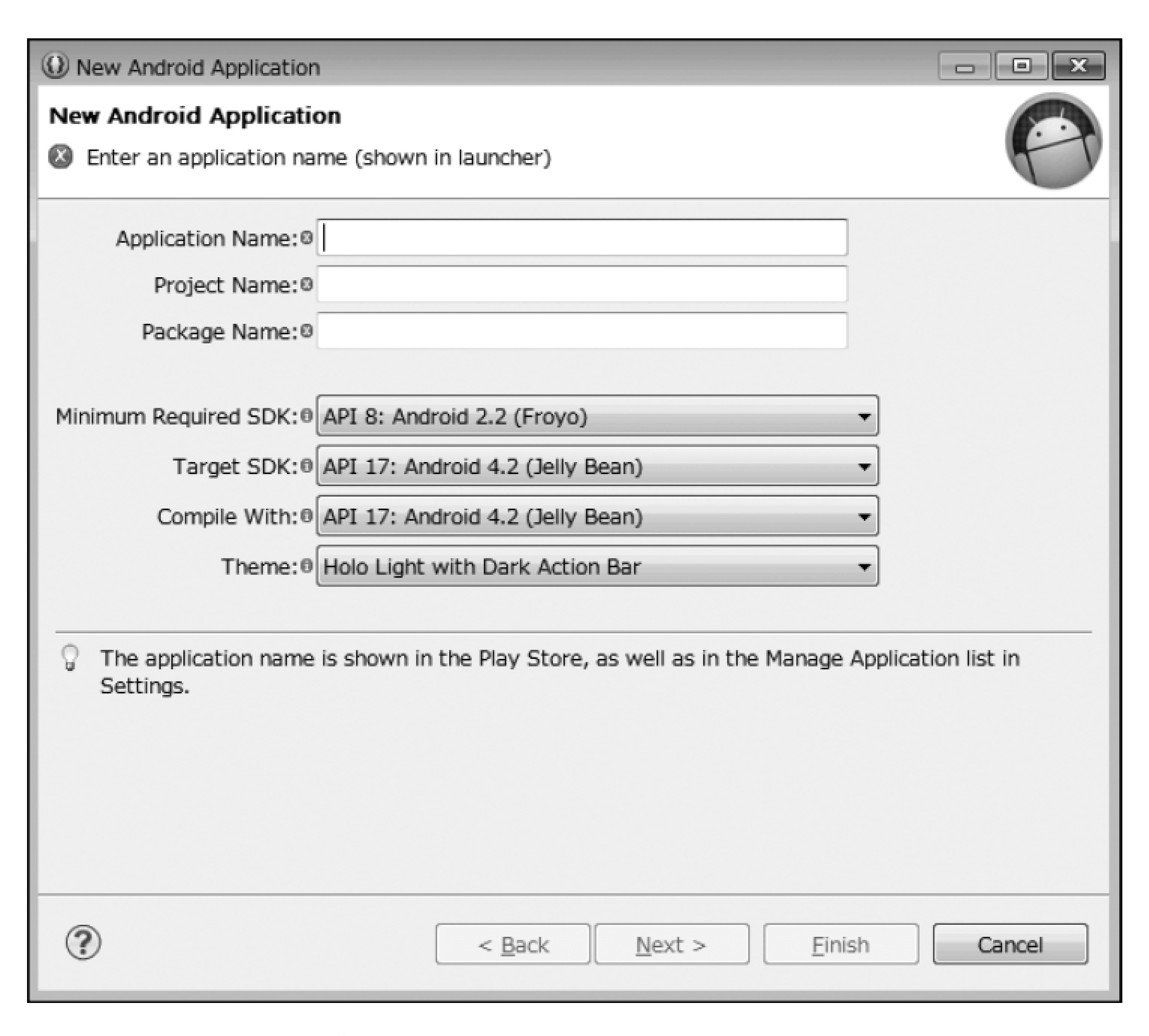


图 2-25　New Android Application

（2）项目名称为 HelloWorld，包名为 cn. jerry，如图 2-26 所示。

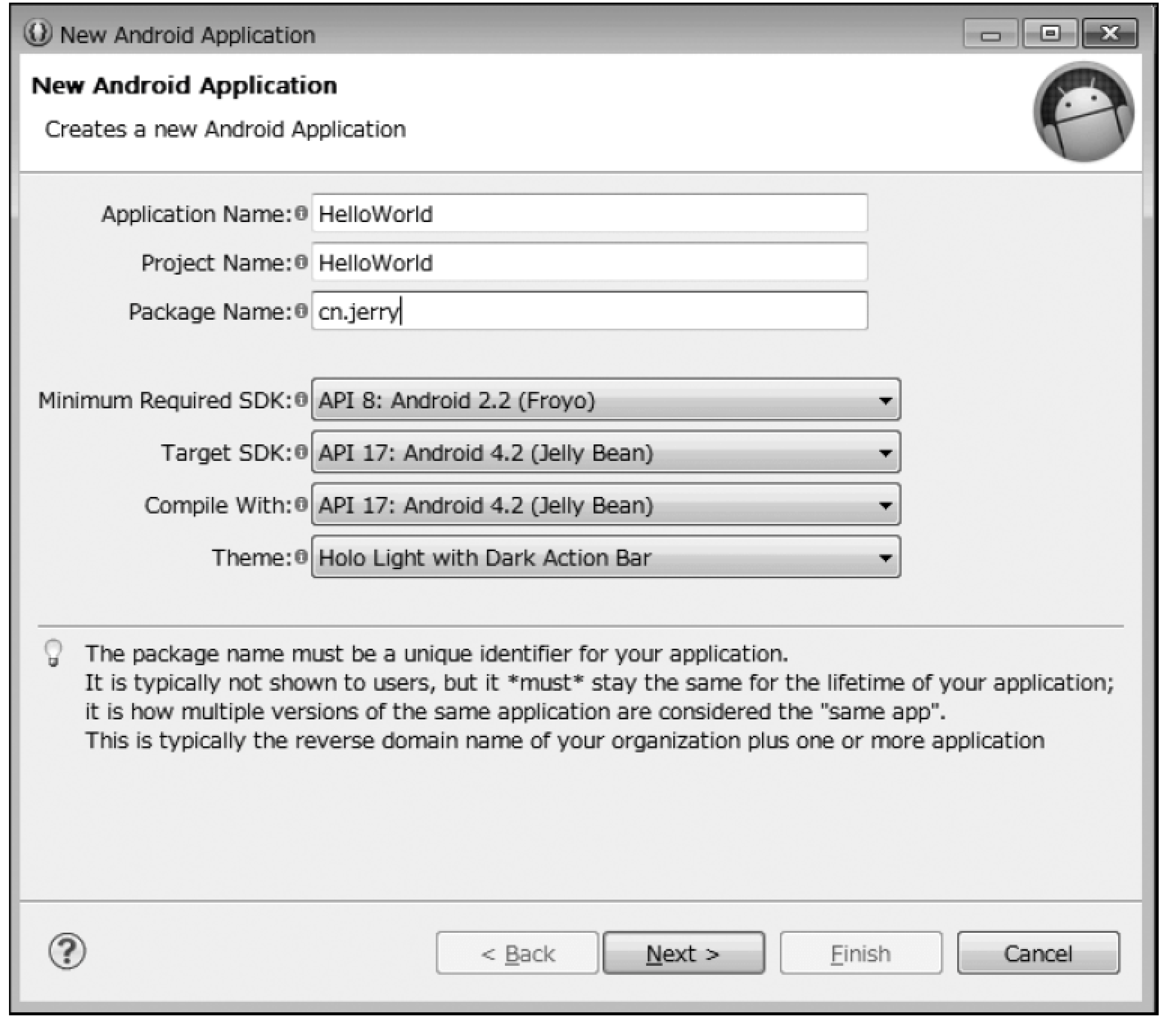


图 2-26　项目命名

(3) 一直单击 Next 按钮到如图 2-27 所示界面，单击 Finish 按钮，创建完成，如图 2-28 所示。

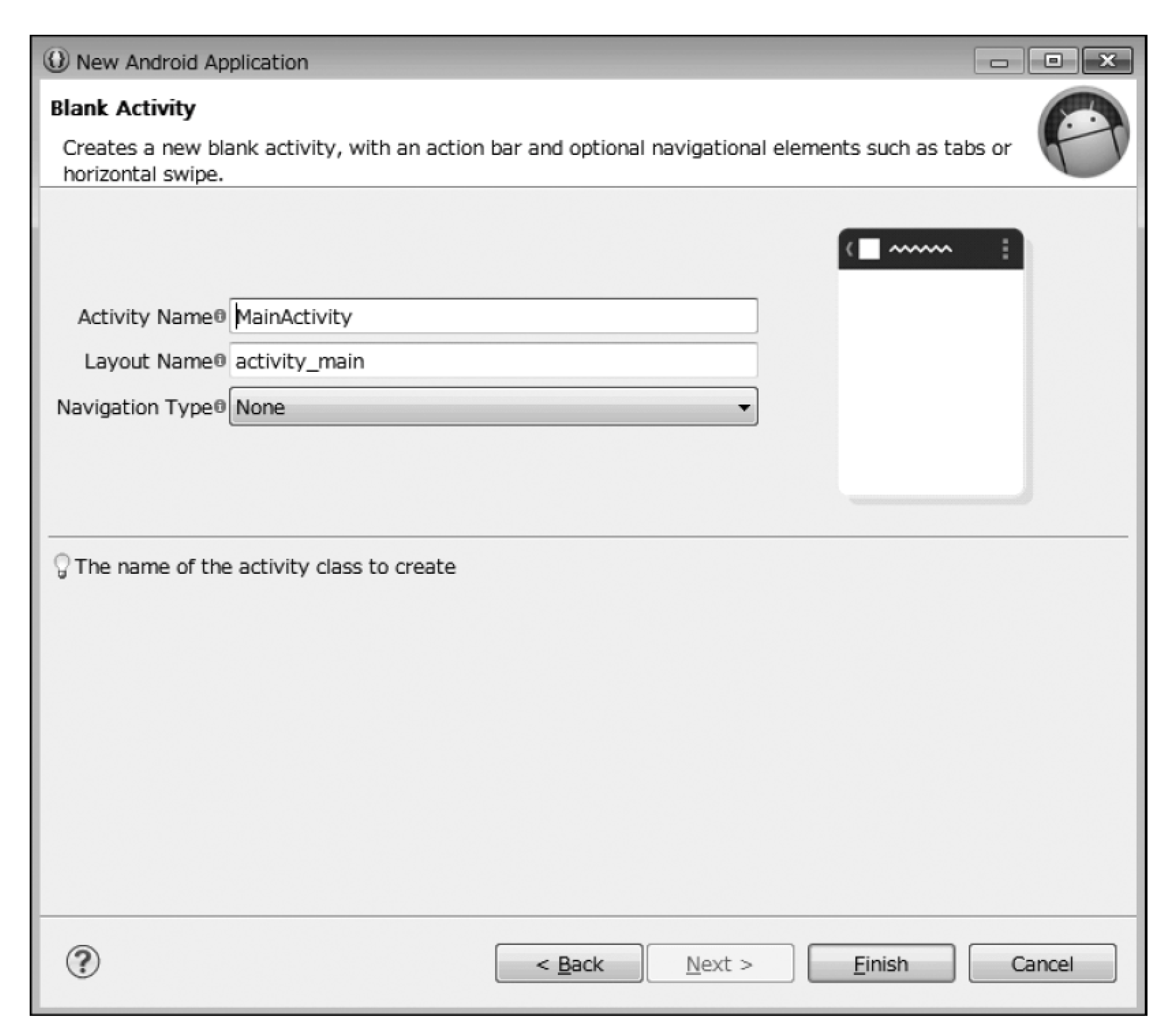


图 2-27　单击 Next 按钮

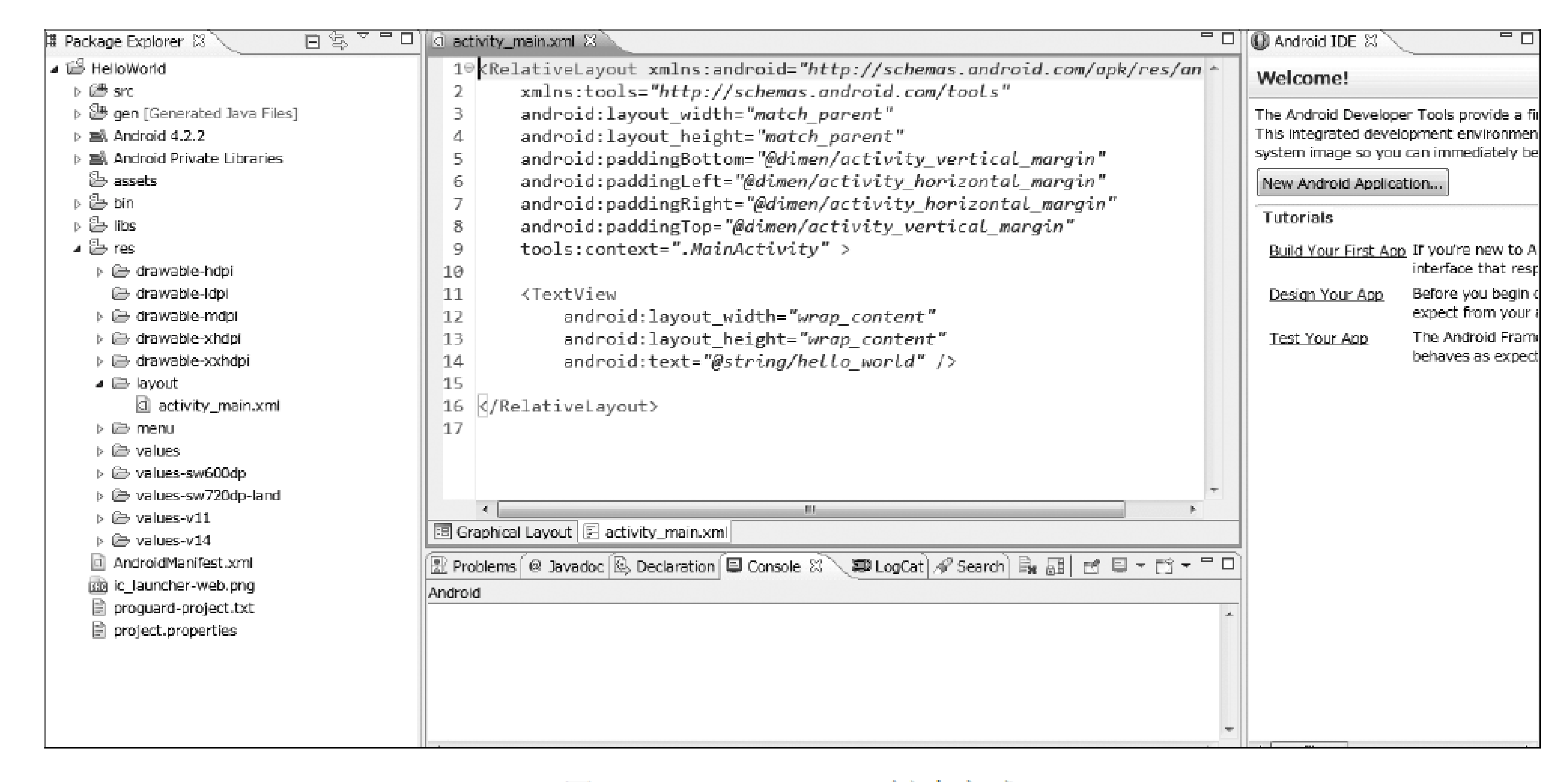


图 2-28　HelloWorld 创建完成

（4）运行程序如图 2-29 所示。

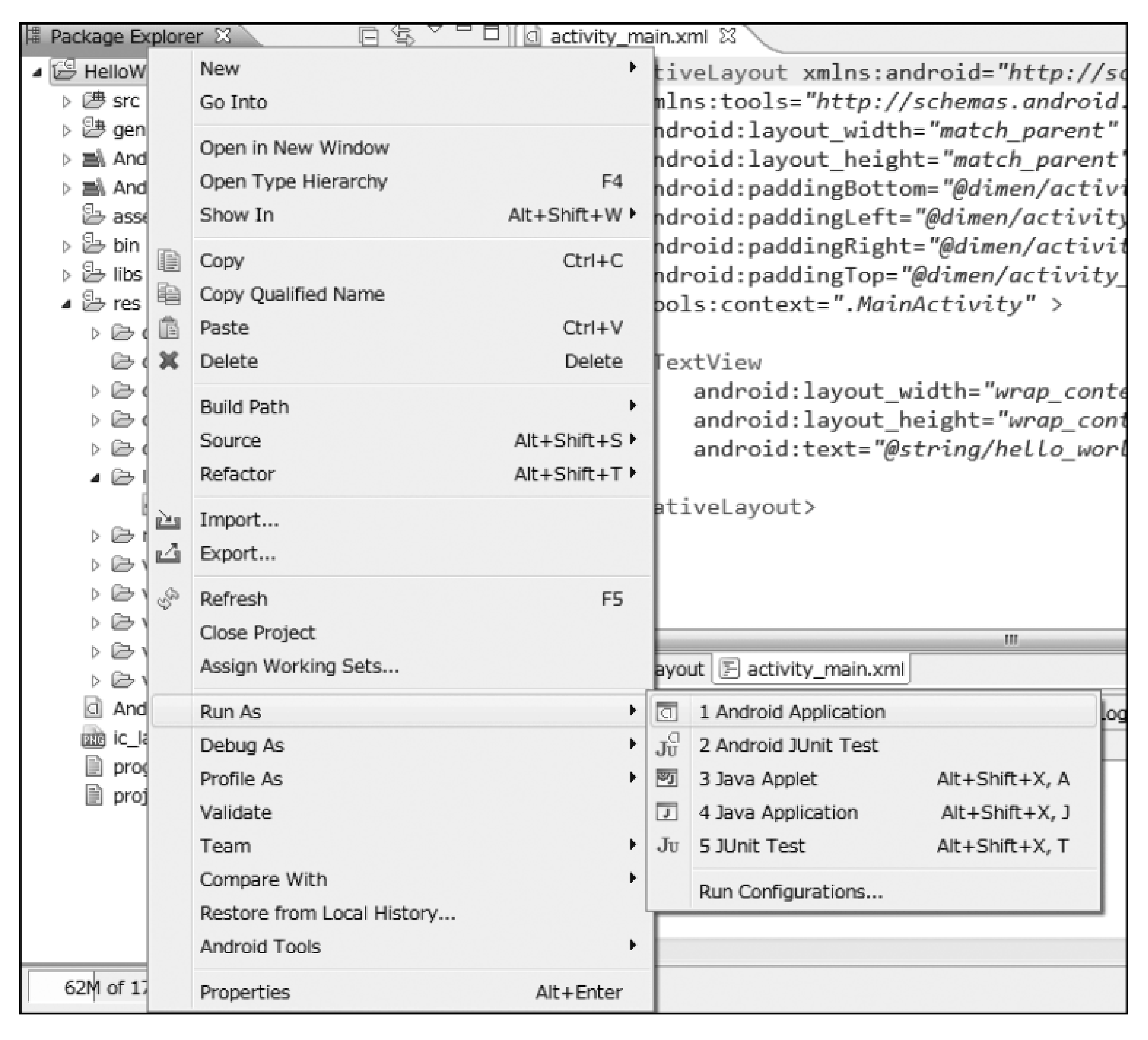


图 2-29 运行程序

（5）运行界面如图 2-30 所示。

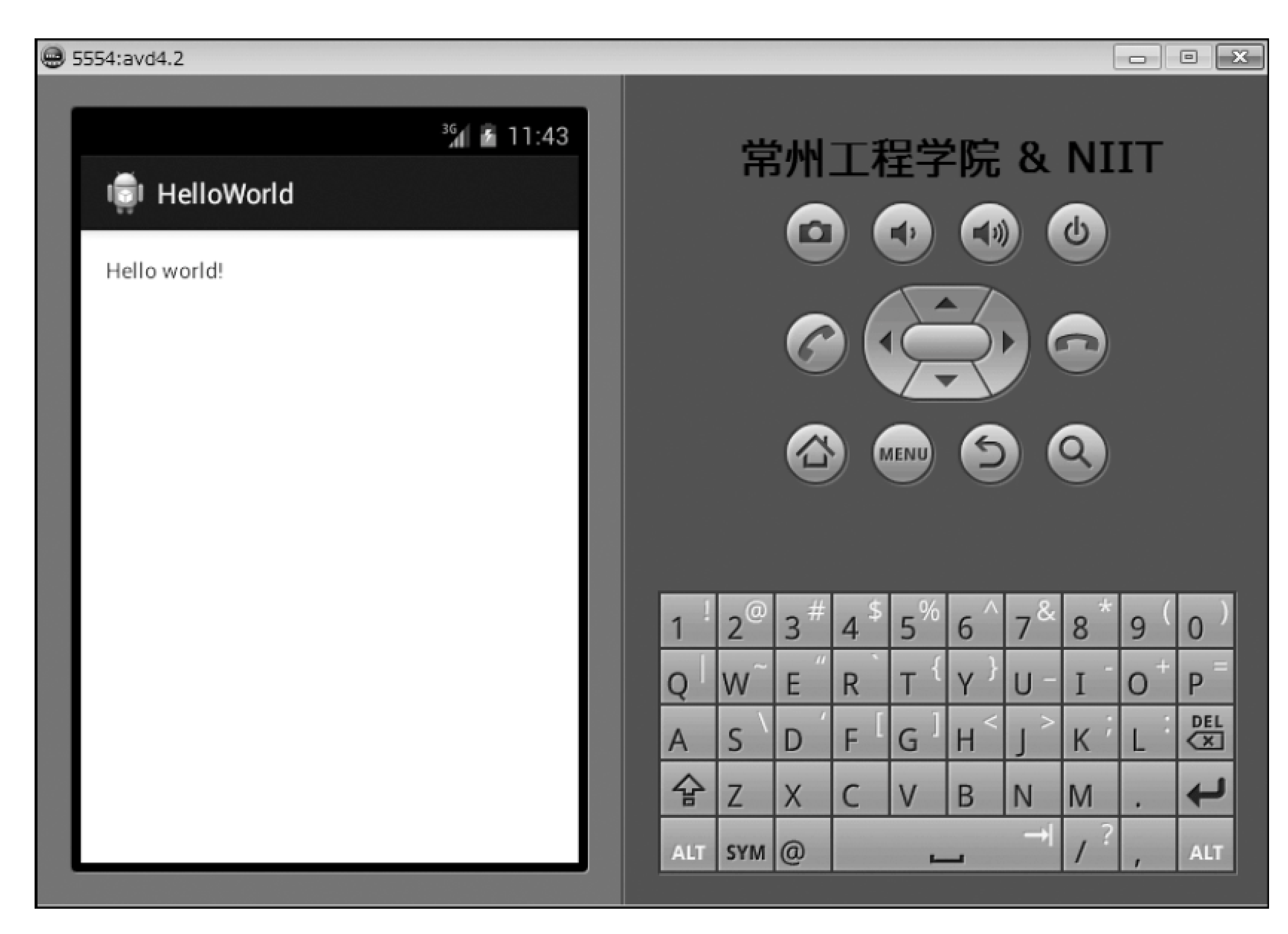


图 2-30 运行界面

2.3 解析 Android 应用程序框架

2.3.1 Android SDK 目录详解

Android SDK 中包含多个文件夹，如图 2-31 所示。它们的作用如下。

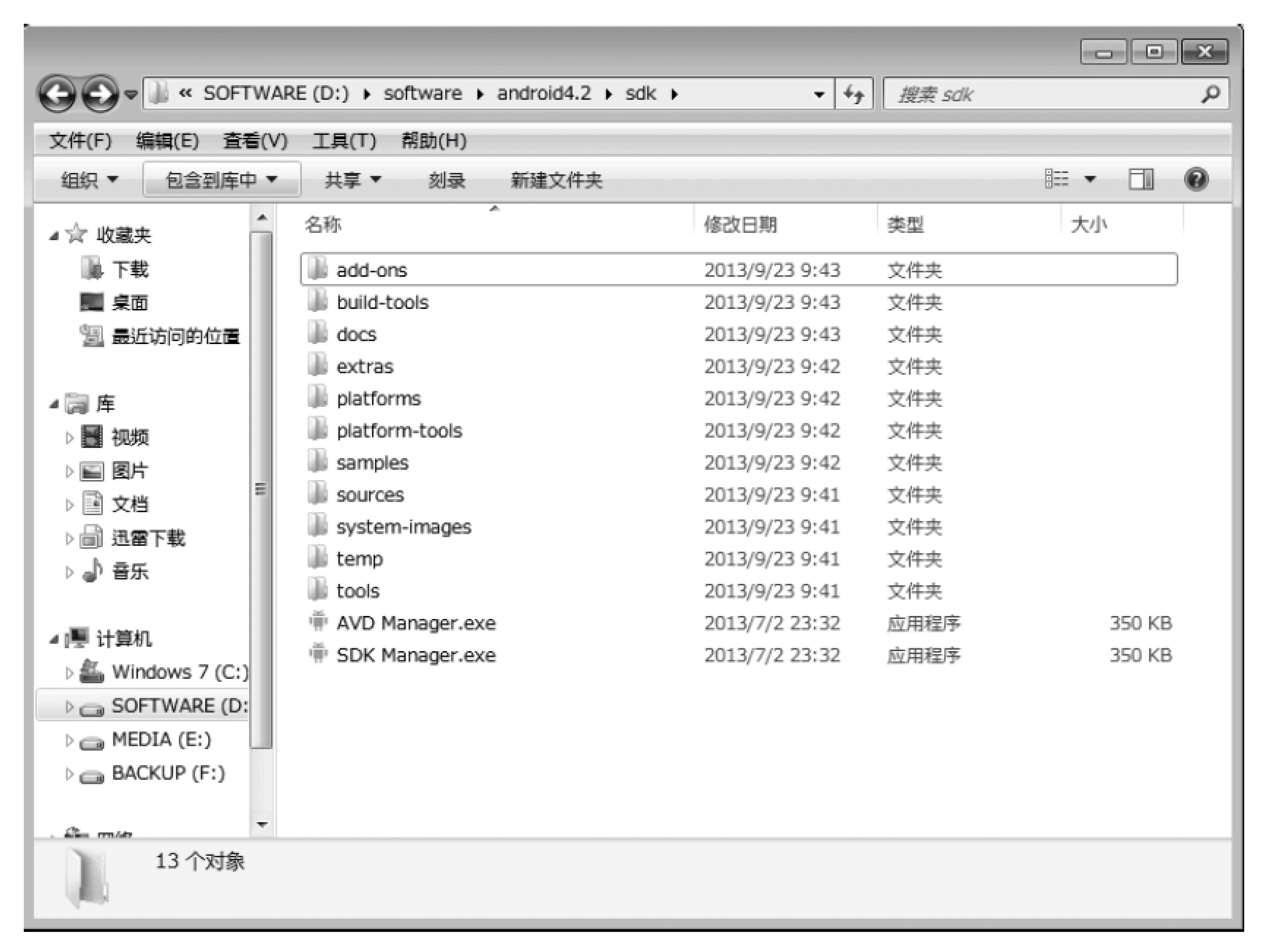


图 2-31 Android SDK 目录

（1）add-ons 保存附加库，比如 GoogleMaps。如果安装了 Ophone SDK，这里也会有一些类库。

（2）build-tools 是编译工具目录，包含了转化为 davlik 虚拟机的编译工具。

（3）docs 是 Android SDK API 参考文档，所有的 API 都可以在这里查到。

（4）extras 是扩展插件目录。

（5）platforms 是每个平台的 SDK 真正的文件，以 Android 2.2 为例，进入后有一个 android-8 的文件夹。android-8 进入后是 Android 2.2 SDK 的主要文件，其中 ant 为 ant 编译脚本，data 保存一些系统资源，images 是模拟器映像文件，skins 是 Android 模拟器的皮肤，templates 是工程创建的默认模板，android.jar 是该版本的主要 framework 文件，tools 目录包含了重要的编译工具，比如 aapt、aidl、逆向调试工具 dexdump 和编译脚本 dx。

（6）platform-tools 保存一些通用工具，比如 adb、aapt、aidl、dx 等文件，Android 123 提示，这里和 platforms 目录中 tools 文件夹有些重复，主要是从 Android 2.3 开始这些工具被划分为通用工具。

(7) samples 是 Android SDK 自带的默认示例工程，apidemos 强烈推荐初学者运行学习。对于 SQLite 数据库操作可以查看 NotePad 这个例子，对于游戏开发 Snake、LunarLander 都是不错的例子，对于 Android 主题开发 Home 是 Android 5.0 时代的主题设计原理。

(8) sources 放的是相关版本的源代码。

(9) system-images 目录是编译好的系统映像，模拟器可以直接加载。

(10) temp 是临时目录。

(11) tools 作为 SDK 根目录下的文件夹，包含了重要的工具，如 ddms 用于启动 Android 调试工具，logcat、屏幕截图和文件管理器；draw9patch 是绘制 android 平台的可缩放 png 图片的工具；sqlite3 可以在 PC 上操作 SQLite 数据库；monkeyrunner 是一个不错的压力测试应用，模拟用户随机按键；mksdcard 是模拟器 SD 映像的创建工具；emulator 是 Android SDK 模拟器主程序，从 Android 1.5 开始，需要输入合适的参数才能启动模拟器；traceview 是 Android 平台上重要的调试工具。

(12) AVD Manager.exe 是虚拟机管理工具。

(13) SDK Manager.exe 是 SDK 管理工具。

2.3.2 Android程序目录结构详解

Android 程序目录结构如图 2-32 所示。

(1) src 文件夹。该文件夹用于存放项目的源代码。

(2) gen 文件夹。该文件夹下有 R.java 文件，R.java 在建立项目时自动生成，是只读模式的，不能更改。R.java 文件中定义了一个类——R，R 类中包含很多静态类，且静态类的名字都与 res 中的一个名字对应，即 R 类定义该项目所有资源的索引。HelloWorld 项目就是如此，如图 2-33 所示。

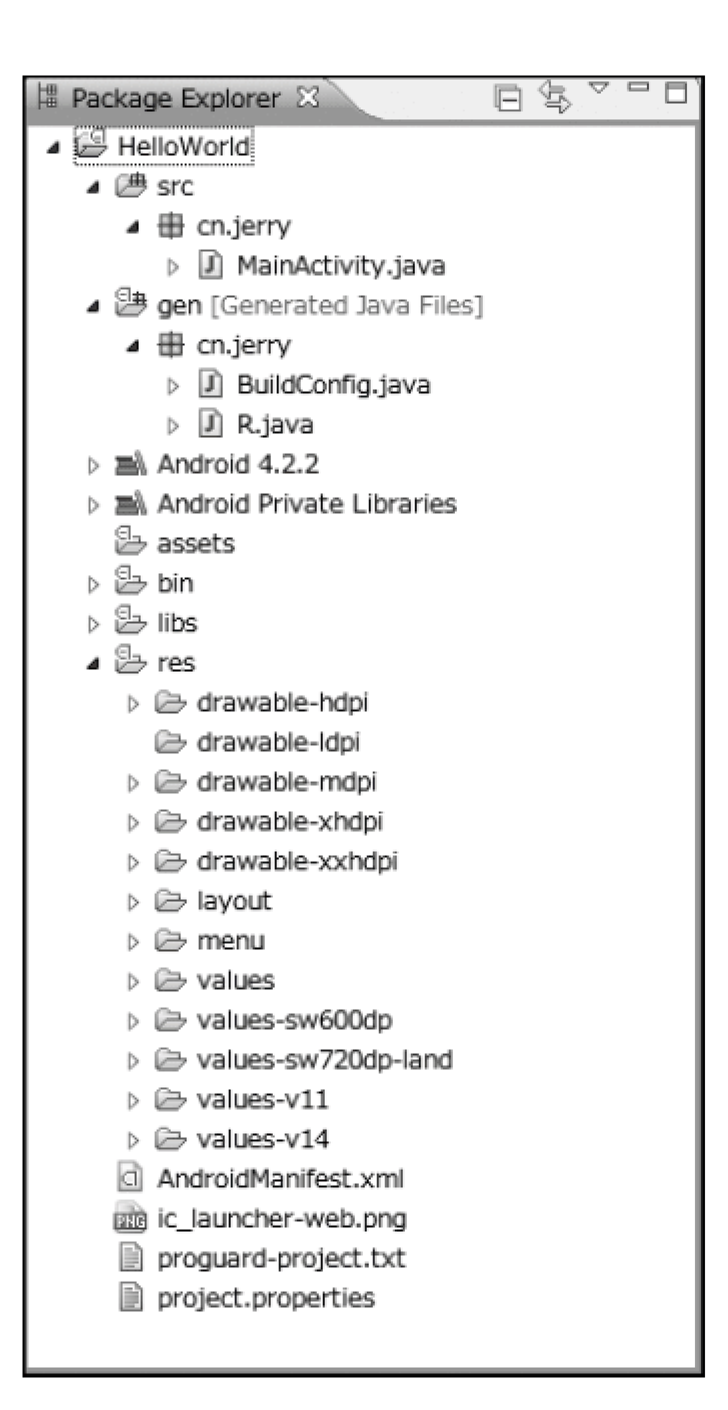


图 2-32　Android 程序目录结构

通过 R.java 可以很快地查找需要的资源，另外编译器也会检查 R.java 列表中的资源是否被使用，没有被使用的资源不会编译软件中，这样可以减少应用在手机中占用的空间。

(3) Android 文件夹。该文件夹下包含 android.jar 文件，这是一个 Java 归档文件，其中包含构建应用程序所需的所有的 Android SDK 库（如 Views、Controls）和 APIs。通过 android.jar 将自己的应用程序绑定到 Android SDK 和 Android Emulator，这允许用户使用所有 Android 的库和包，且使应用程序在适当的环境中调试。例如，上面的 HelloWorld.java 源文件中的：

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.

package helloworld.test;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

```
res
  drawable-hdpi
    icon.png
  drawable-ldpi
    icon.png
  drawable-mdpi
    icon.png
  layout
    main.xml
  values
    strings.xml
```

图 2-33　gen 文件夹

① import android. app. Activity。

② import android. os. Bundle。

这里两行代码就是从 android. jar 导入的。

(4) assets 文件夹。包含应用系统需要使用的诸如 MP3、视频类的文件。

(5) res 文件夹。资源目录，包含项目中的资源文件并将其编译应用程序。向此目录添加资源时，会被 R. java 自动记录。新建一个项目，res 目录下会有三个子目录：drawabel、layout、values。

① drawabel。包含一些应用程序可以用的图标文件(* . png、* . jpg)。

② layout。界面布局文件(main. xml)与 Web 应用中的 HTML 类同，没修改过的 main. xml 文件如下(HelloWorld 没有修改过)。

XML/HTML 代码如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
</RelativeLayout>
```

③ values。软件上所需要显示的各种文字。可以存放多个 * . xml 文件，还可以存放不同类型的数据。比如 arrays. xml、colors. xml、dimens. xml、styles. xml。

(6) AndroidManifest. xml。项目的总配置文件，记录应用中所使用的各种组件。这

个文件列出了应用程序所提供的功能，在这个文件中，可以指定应用程序使用的服务(如电话服务、互联网服务、短信服务、GPS服务等)。另外，当新添加一个Activity，也需要在这个文件中进行相应配置，只有配置好后，才能调用此Activity。AndroidManifest.xml将包含application permissions、Activities、intent filters等的设置。

HelloWorld项目的AndroidManifest.xml如下所示。

XML/HTML代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="cn.jerry"
    android:versionCode="1"
    android:versionName="1.0">
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme">
        <activity
            android:name="cn.jerry.MainActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

(7) project.properties。记录项目中所需要的环境信息，比如Android的版本等。HelloWorld的project.properties文件代码如下所示，代码中的注释已经把project.properties解释得很清楚，最后一行指定了本项目所用的sdk版本号。

```
#This file is automatically generated by Android Tools.
#Do not modify this file--YOUR CHANGES WILL BE ERASED!
#
#This file must be checked in Version Control Systems.
#
#To customize properties used by the Ant build system edit
#"ant.properties", and override values to adapt the script to your
#project structure.
#
```

```
#To enable ProGuard to shrink and obfuscate your code, uncomment this (available properties: sdk.dir,
user.home):
#proguard.config=${sdk.dir}/tools/proguard/proguard-android.txt:proguard-
project.txt

#Project target.
target=android-17
```

2.4 本章小结

本章主要介绍了搭建Android开发环境的步骤、Android SDK的相关内容、应用程序的结构,并创建了一个简单的Android应用程序。通过本章的学习,读者可以掌握Android开发环境的配置,比较清晰地把握Android SDK的全貌,熟悉应用程序的结构,以及附带的工具使用。

第 3 章

通信功能的设计及开发

本章的工作目标如下:

(1) 使用 Android 中布局和基本控件实现打电话、发送短信界面。

(2) 使用 Activity 和 Intent 实现拨打电话、发送短信功能。

(3) 完成程序运行与效果测试。

3.1 项目分析

本项目的学习主要关注以下几点。

(1) 认识 Android 中视图的层次结构。

(2) 掌握 Android 中常用布局。

(3) 掌握 Android 中标签、文本框、按钮控件。

(4) 熟悉 Android 中 Intent(意图)的使用。

(5) 掌握 Android 中 Activity 的使用。

(6) 了解 Android 中 permission 的概念。

(7) 综合前面几点实现拨打电话和发送短信功能。

通信功能的设计及开发如图 3-1 所示。

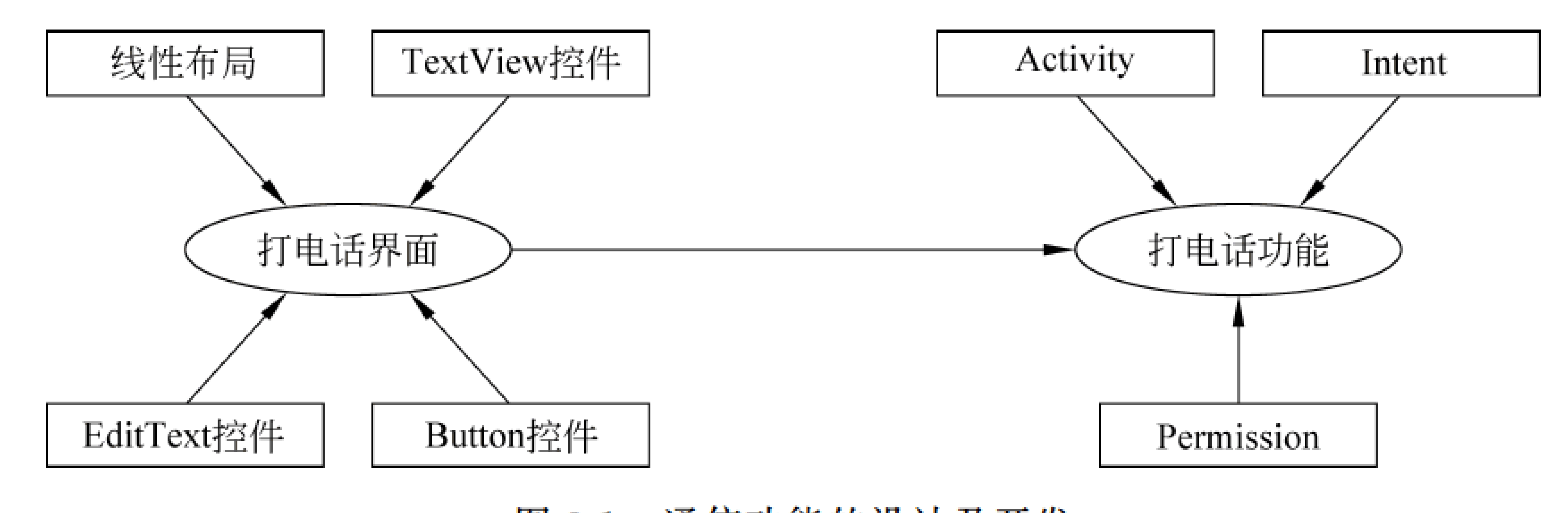


图 3-1 通信功能的设计及开发

1. 打电话界面编写

首先,在界面中加入垂直方向的线性布局,让控件按照从上到下的方式排列。其次,在界面中依次加入 TextView 标签控件显示“请输入手机号码”,加入 EditText 文本框控

件，让用户输入手机号码，加入 Button 按钮控件，显示“拨打”。

2. 打电话功能编写

首先，新建一个 Activity，调用 Activity 的 setContentView()方法与界面建立关联。其次，通过 Intent 意图启动拨打电话功能，最后，在 AndroidManifest. xml 文件中加入 Permission 权限。

3.2 项目界面设计

Android 用户界面的开发包括两个方面：用户界面设计和相应的事件处理。在一个 Android 应用程序中，用户界面由一系列的 View 和 ViewGroup 对象组合而成。Android 有 View 和 ViewGroup 对象，它们都继承自 View 基类；事件处理则包括 button 控件的单击事件等。

1. 用户界面的生成

Android 用户界面的生成有两种：一种是通过 XML 布局文件生成，这也是最简便和最直观的一种方法；另一种是用代码直接生成。对于 XML 生成的布局文件可以通过 ADT 提供的 UI 预览功能预览所创建的用户界面。

2. View

Android 中的 View 与以前理解的“视图”不同。在 Android 中，View 比视图具有更广的含义，它包含用户交互和显示，更像 Windows 操作系统中的 Window。View 对象是 Android 平台中用户界面体现的基础单位。View 类是其称为“Widget（工具）”的子类的基础，它们提供了诸如文本输入框和按钮之类的 UI 对象的完整实现。一个 View 对象是一个数据结构，存储布局参数和屏幕特定区矩形区域的内容。一个 View 会处理自己所在屏幕区域的测量、布局、绘制、焦点改变、滚动和按键手势交互。作为用户交互对象，一个 View 可以作为用户与系统的交互工具，接收事件。作为基类，View 类为 Widget 服务，Widget 是一组用于绘制交互屏幕元素的完全实现子类。Widget 处理自己的测距和绘图，所以可以快速地用它们去构建界面。常用的 Widget 包括 TextView、EditText 及 Button 等。

3. ViewGroup

ViewGroup 是 View 的子类，它具有 View 特性，但它主要用来充当 View 的容器，将其中的 View 视作自己的孩子，对它的子 View 进行管理，当然它的孩子也可以是 ViewGroup 类型。ViewGroup 和它的孩子们（View 和 ViewGroup）形成了一个树形结构，View 类有接收和处理消息的功能，Android 系统所产生的消息会在这些 ViewGroup 和 View 之间传递。

ViewGroup 可以为界面增加结构，并且将复杂的屏幕元素组成一个独立的实体。

ViewGroup 为 Layout 提供服务，Layout 用来提供各种布局结构，包括 linear 线性布局、表格布局和绝对布局等。

图 3-2 所示是一个由 View 和 ViewGroup 布局的活动(Activity)界面。一个 Activity 的界面可以包含多个 ViewGroup 和 View，通过两者的组合使用能够更好地完成更复杂界面的设计。

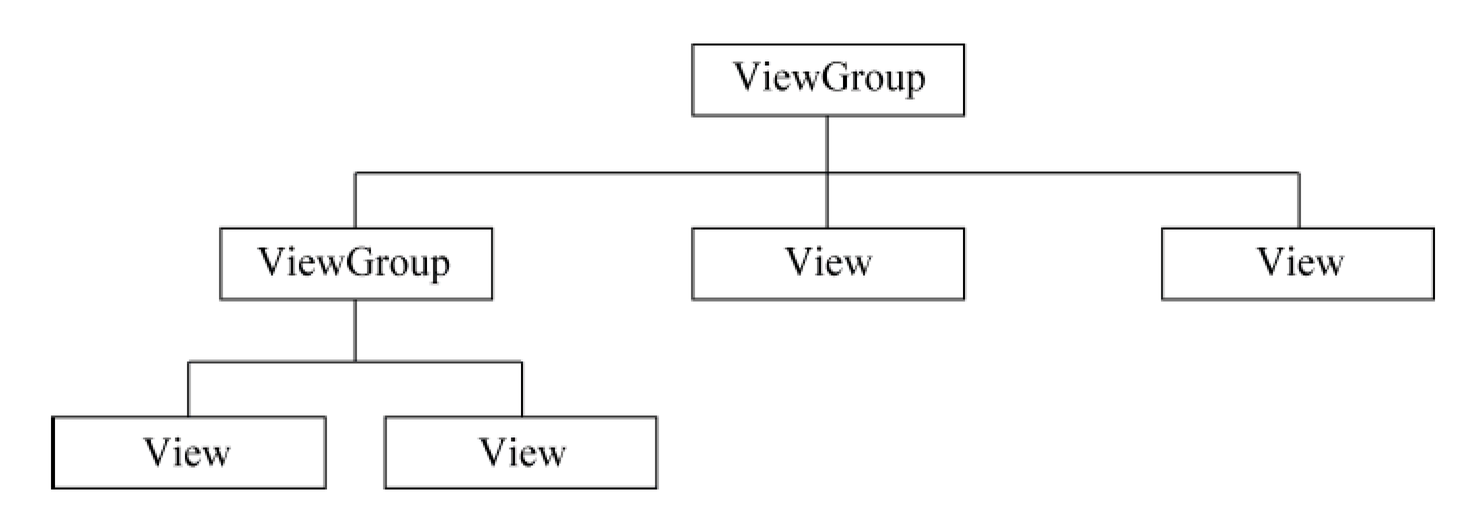


图 3-2　View 与 ViewGroup 组合使用布局的 Activity 界面

一个新的 Activity 被创建时是一个空白屏幕，可以把自己的用户界面放到上面。要设置用户界面，可以调用 setContentView()，并传入需要显示的 View 实例(通常是一个布局)。由于空白屏幕不是我们想要的，所以在创建一个新的 Activity 时，在 onCreate() 处理程序中总是采用 setContentView()的方法设置我们需要显示的用户界面。具体使用 Activity，请参照本章相关实例。

3.2.1　知识准备

1. 布局

Android 的界面是由布局和组件协同完成的，布局好比是建筑里的框架，而组件则相当于建筑里的砖瓦。组件按照布局的要求依次排列，就组成了用户所看见的界面。Android 的五大布局分别是 RelativeLayout(相对布局)、LinearLayout(线性布局)、FrameLayout(单帧布局)、TableLayout(表格布局)和 AbsoluteLayout(绝对布局)。前两种布局在项目实战中使用较多，后两种布局在项目中使用较少，本书所有项目重点使用的是前两种布局，在 QQ 项目中使用 FrameLayout 布局。

(1) RelativeLayout

RelativeLayout 按照各子元素之间的位置关系完成布局。在此布局中的子元素里，与位置相关的属性将生效。例如，android: layout_below、android:layout_above 等。子元素就通过这些属性和各自的 ID 配合指定位置关系。在指定位置关系时，引用的 ID 必须在引用之前先被定义，否则将出现异常。不能在 RelativeLayout 容器本身和它的子元素之间产生循环依赖，比如，不能将 RelativeLayout 的高设置成 WRAP_CONTENT 时将子元素的高设置成 ALIGN_PARENT_BOTTOM。

RelativeLayout 常用的位置属性如下。

- android:layout_above：将该控件置于给定 ID 的控件之上。
- android:layout_below：将该控件置于给定 ID 控件之下。

- android:layout_toLeftOf：将该控件置于给定 ID 的控件之左。
- android:layout_toRightOf：将该控件置于给定 ID 的控件之右。
- android:layout_alignBaseline：该控件基线对齐给定 ID 的基线。
- android:layout_alignBottom：该控件于给定 ID 的控件底部对齐。
- android:layout_alignLeft：该控件于给定 ID 的控件左对齐。
- android:layout_alignRight：该控件于给定 ID 的控件右对齐。
- android:layout_alignTop：该控件于给定 ID 的控件顶对齐。
- android:layout_alignParentLeft：如果为 True，该控件位于父控件的左部。
- android:layout_alignParentRight：如果为 True，该控件位于父控件的右部。
- android:layout_alignParentTop：如果为 True，该控件位于父控件的顶部。
- android:layout_alignParentBottom：如果为 True，该控件位于父控件的底部。
- android:layout_centerHorizontal：如果为 True，该控件将被置于水平方向的中央。
- android:layout_centerInParent：如果为 True，该控件将被置于父控件水平方向和垂直方向。
- android:layout_centerVertical：如果为 True，该控件将被置于垂直方向的中央。

RelativeLayout 是 Android 五大布局结构中最灵活的一种布局结构，比较适合一些复杂界面的布局，效果如图 3-3 所示。



图 3-3　RelativeLayout 布局

开发步骤。

第一步：新建 Android 工程。在 eclipse 左上角单击 File→New→Android Application Project，在弹出的对话框中填写应用名称（Application Name）、项目名称（Project Name）和包名（Package Name），如图 3-4 所示。

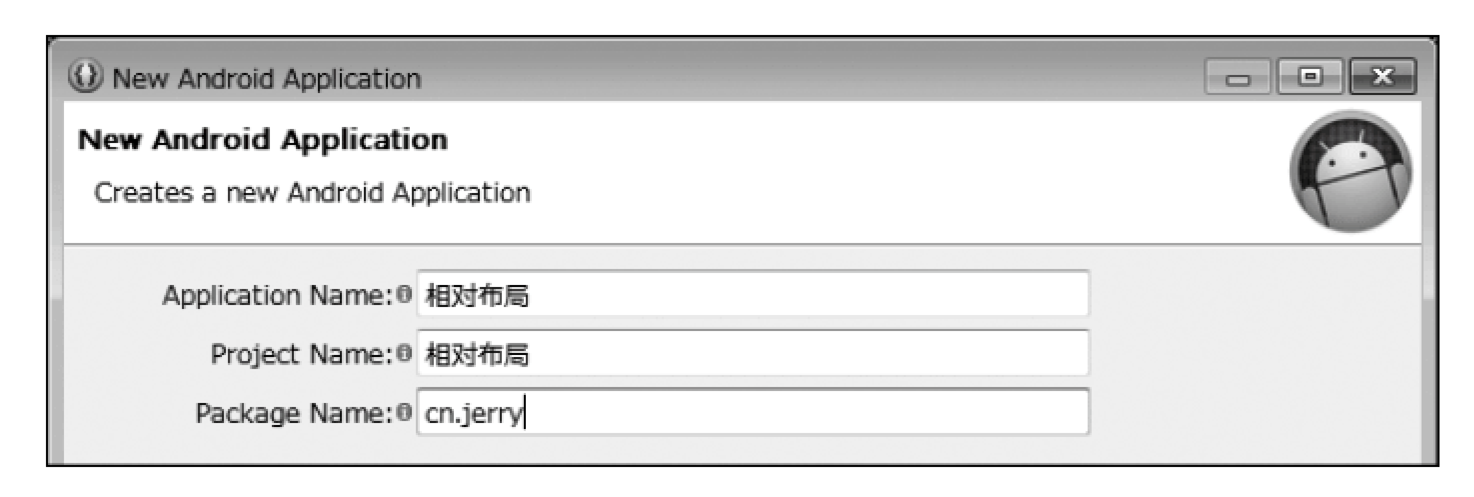


图 3-4　填写相应内容

一直单击 Next 按钮，直到单击 Finish 按钮结束，Android 工程创建成功。

第二步：添加布局代码。在项目中双击 res→layout 目录下的 Activity_main. xml，添加如下代码。

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="请输入用户名：" />
<EditText
    android:id="@+id/editText1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/textView1">
    <requestFocus />
</EditText>
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/editText1"
    android:layout_alignParentRight="true"
    android:text="OK"
     />
<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/editText1"
    android:layout_toLeftOf="@id/button1"
    android:layout_marginRight="20px"
    android:text="Cancle" />
```

代码解释如下。

① android:layout_below="@id/textView1"：这行代码指定文本框在 id 为 textView1 的 TextView 控件下方。

② android:layout_alignParentRight="true"：这行代码指定按钮位于手机屏幕的右边。

③ android:layout_toLeftOf="@id/button1"：这行代码指定 Button 按钮位于 id 为

button2 的按钮位于 id 为 button1 的按钮左边。

单击 Graphical Layout 切换到布局视图界面，可以看到界面效果。

第三步：运行程序。选中项目，右击选择 Run as 命令，单击框图部分（Android Application）运行后可以在模拟器上看到运行效果。

（2）LinearLayout

LinearLayout 按照垂直或者水平的顺序依次排列子元素，通过 android:orientation 属性可以设置线性布局的方向，每一个子元素都位于前一个元素之后。如果是垂直排列，则是一个 *N* 行单列的结构，每一行只会有一个元素，而不论这个元素的宽度为多少；如果是水平排列，则是一个单行 *N* 列的结构。如果搭建两行两列的结构，通常的方式是先垂直排列两个元素，每一个元素里再包含一个 LinearLayout 进行水平排列。

LinearLayout 中的子元素的常用属性如下。

- android:id：为控件指定相应的 ID。
- android:text：指定控件中显示的文字，需要注意的是，这里尽量使用 strings. xml 文件中的字符串。
- android:grivity：指定控件的基本位置，比如居中、居右等位置。
- android:textSize：指定控件中字体的大小。
- android:background：指定该控件所使用的背景色，RGB 命名法。
- android:width：指定控件的宽度。
- android:height：指定控件的高度。
- android:padding*：指定控件的内边距，也就是说控件中的内容。
- android:sigleLine：如果设置为真，则将控件的内容在同一行中进行显示。

线性布局是程序中最常见的一种布局方式，以下实例介绍了线性布局的使用，效果如图 3-5 所示。



图 3-5 线性布局

开发步骤。

第一步：新建项目“线性布局 1”，在 res→layout 目录下新建布局文件 main.xml。

第二步：在弹出的对话框中输入文件名 main，选择 xml 文件根节点 LinearLayout，单击 Finish 按钮，如图 3-6 所示。



图 3-6　填写相应内容

在 main.xml 文件中添加以下代码。

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="200dp"
    android:text="第一行"
    android:background="#FF0000"
    android:layout_weight="2"
    android:gravity="center_vertical"
    android:textSize="50sp"
    android:paddingLeft="20px"
    />
<TextView
    android:layout_width="fill_parent"
    android:layout_height="100dp"
    android:text="第二行"
    android:background="#0000FF"
    android:layout_weight="1"
    android:gravity="center_vertical"
    android:textSize="30sp"
    />
```

代码解释如下。

① android:gravity="center_vertical":这行代码设置 TextView 控件的 2 文本水平居中。

② android:layout_weight="1":这行代码设置 TextView 占整个屏幕的权重为 1,占整个幕的 1/3。

第三步:删除 layout 下 Activity_mail.xml 文件,打开 MianActivity.java。

第四步:修改 java 文件对应的布局,修改的 java 代码如下:setContentView(R.layout.main)。

2. TextView 控件的使用

TextView 控件能向用户展现文本信息(包括 HTML 文本),可以自己设置该文本信息是否能够编辑。

(1) 如何简单通过 XML 布局使用 TextView。

第一步:打开布局文件 main.xml,如图 3-7 所示。

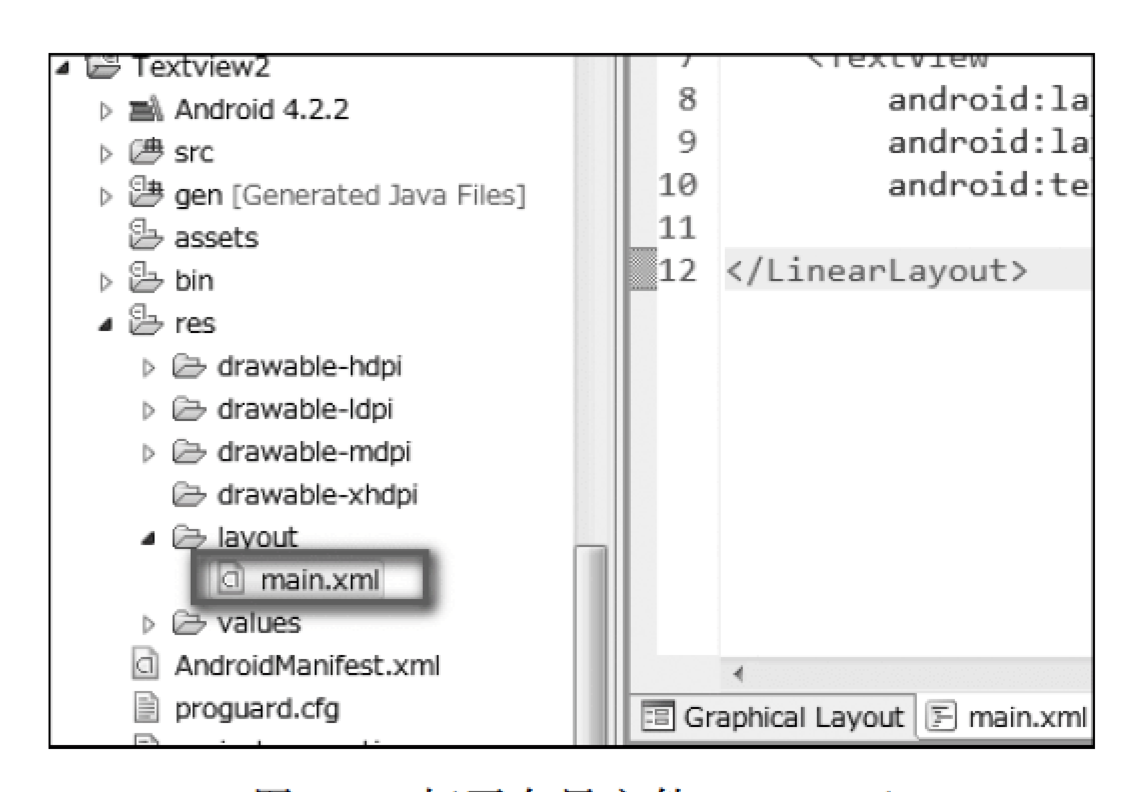


图 3-7 打开布局文件 main.xml

第二步:在 XML 布局文件中使用以下代码。

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/username" />
```

代码解释如下。

① layout_width:设置 TextView 控件的宽度,系统提供 3 个值,分别是 fill_parent、wrap_content 和 match_parent。其中,wrap_content 表示大小刚好足够显示当前控件里的内容。

Android 中 fill_parent 和 match_parent(从 Android 2.2)是一样的。为了兼容低版本,建议使用 fill_parent。

设置布局/控件为 fill_parent 将强制性让它布满整个屏幕或填满父控件的空白。

② layout_height:设置 TextView 控件的高度。

③ text:设置 TextView 控件的文本值。

第三步:运行程序,如图 3-8 所示。



图 3-8　运行程序

(2) 如何简单通过代码使用 TextView。

第一步：打开 TextView1_Activity.java。

第二步：在程序中创建 TextView 对象的代码如下。

```
//新建标签对象
TextView tv=new TextView(this);
//为标签对象赋值
tv.setText("周杰伦");
//把标签绑定到 Acivtity 中
setContentView(tv);
```

第三步：程序运行如图 3-9 所示。



图 3-9　创建 TextView 对象

(3) TextView 常用属性。

TextView 常用属性如下。

① Android 中的颜色设置。

```
android:textColor= "#F8FF00";
```

设置文本颜色,F8FF00 是 RGB 表示方法。

• 利用系统自带的颜色类。

```
tx.setTextColor(android.graphics.Color.RED);
```

• 数字颜色表示。

```
tx.setTextColor(0xffff00ff);
```

• 直接在 xml 的 TextView 中设置。

```
android:textColor= "# F8F8FF00";
```

或

```
android:textColor= "#F8FF00";
```

说明:

• 0xffff00ff 是 int 类型的数据,分组 0x|ff|ff00ff,0x 表示颜色整数的标记,ff 表示透明度,ff00ff 表示色值,注意:0x 后面 ffff00ff 必须是 8 位的颜色表示。
• 颜色和不透明度(alpha)值以十六进制表示法表示。任何一种颜色的值范围都是 0~255(00~ff)。
• 对于 alpha,00 表示完全透明,ff 表示完全不透明。
• 表达式顺序是 aabbggrr,其中 aa=alpha(00 到 ff);bb=blue(00 到 ff);gg=green(00 到 ff);rr=red(00 到 ff)。

效果如图 3-10 所示。



图 3-10 Android 中的颜色设置

② 设置链接(代码清单：CH3_6_ TextView4)，如图 3-11 所示。

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="http://www.baidu.com;13813888888"
    android:autoLink="all"
    />
```

代码解释如下。

android:autoLink：设置是否当文本为 URL 链接/email/电话号码/map 时，文本显示为可单击的链接，可选值(none/web/email/phone/map/all)。



图 3-11　设置链接

3．EditText 控件的使用

EditText 是接收用户输入信息的控件，如图 3-12 所示。

EditText 常用的属性如下。

- android:maxLength="3"：限制输入字符数量。
- android:singleLine="false"：功能为设置多行文本框。
- android:inputType="number"：功能为限制 EditText 输入信息。
- android:hint="我是 EditText"：设置提示信息。
- android:drawableLeft="@drawable/title"：在 EditText 中显示图片。
- android:background="@drawable/shape"：设置圆角。

设置圆角，如图 3-13 所示，shape.xml 文件代码如下：



图 3-12　EditText



图 3-13　设置圆角

```
< ?xml version= "1.0" encoding= "UTF- 8"?>
< shape
    xmlns:android= "http://schemas.android.com/apk/res/android"
    android:shape= "rectangle">
    < !--填充的颜色-->
    < solid android:color= "#FFFFFF" />
    < !--设置矩形的四个角为弧形-->
```

```
    <!--android:radius 弧形的半径-->
    <corners android:radius="17dip" />
</shape>
```

4. Button 控件的使用

Button 是按钮控件，主要用来产生单击事件。单击事件主要通过监听器实现，下面是几种实现方式。

(1) 定义一个单击监听器属性。

```
OnClickListener listener=new OnClickListener() {
    @Override
    public void onClick(View v) {
        System.out.println("您单击到了这个按钮...");
    }
};
//找到按钮，把按钮与监听器绑定起来
Button button=(Button) findViewById(R.id.button1);
button.setOnClickListener(listener);
```

(2) 通过匿名内部类定义监听器。

```
Button button=(Button) findViewById(R.id.button1);
button.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        System.out.println("您单击到了这个按钮...");

    }
});
```

(3) 通过外部类定义监听器。

```
class MyListener implements OnClickListener{
    /**
     * @param context
     */
    public MyListener(Context context) {
        this.context=context;
    }


    private Context context;


    @Override
    public void onClick(View v) {
        System.out.println("您单击到了这个按钮...");
    }
```

```
}
Button button= (Button) findViewById(R.id.button1);
MyListener listener= new MyListener(Button1Activity.this);
button.setOnClickListener(listener);
```

(4) 在按钮控件中添加 onClick 属性(见图 3-14)。

```
<Button
        android:id= "@ + id/button1"
        android:layout_width= "wrap_content"
        android:layout_height= "wrap_content"
        android:text= "单击"
        android:onClick= "display"
         />
public void display(View v){
    System.out.println("您单击到了这个按钮...");
}
```



图 3-14 Button 控件

3.2.2 项目界面相关代码设计

通信功能的界面设计步骤如下。

(1) 拨打电话界面设计。新建 call 项目,在 res 目录下打开 main.xml 布局文件,该布局文件就是在布局中描述的相对布局(relativelayout),如图 3-15 所示。

对应的 XML 代码如下:

```
<?xml version= "1.0" encoding= "utf-8"?>
<LinearLayout xmlns:android= "http://schemas.android.com/apk/res/android"
```

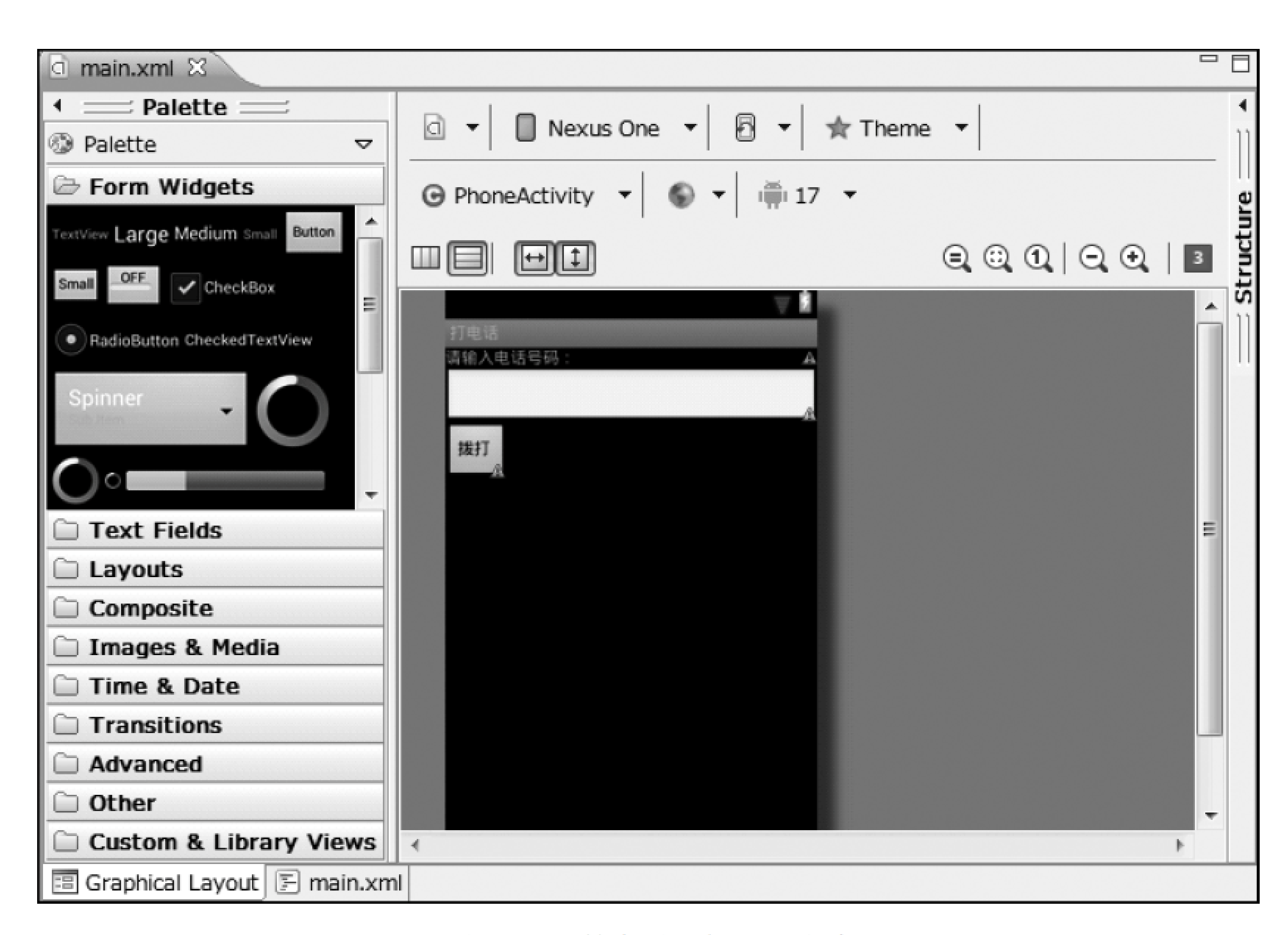


图 3-15　拨打电话界面设计

```
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical">

<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="请输入电话号码:" />

<EditText
    android:id="@+id/et_mobile"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
</EditText>

<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="拨打"
    android:onClick="call"
     />

</LinearLayout>
```

（2）发送短信界面设计，如图 3-16 所示。

对应的 XML 代码如下：

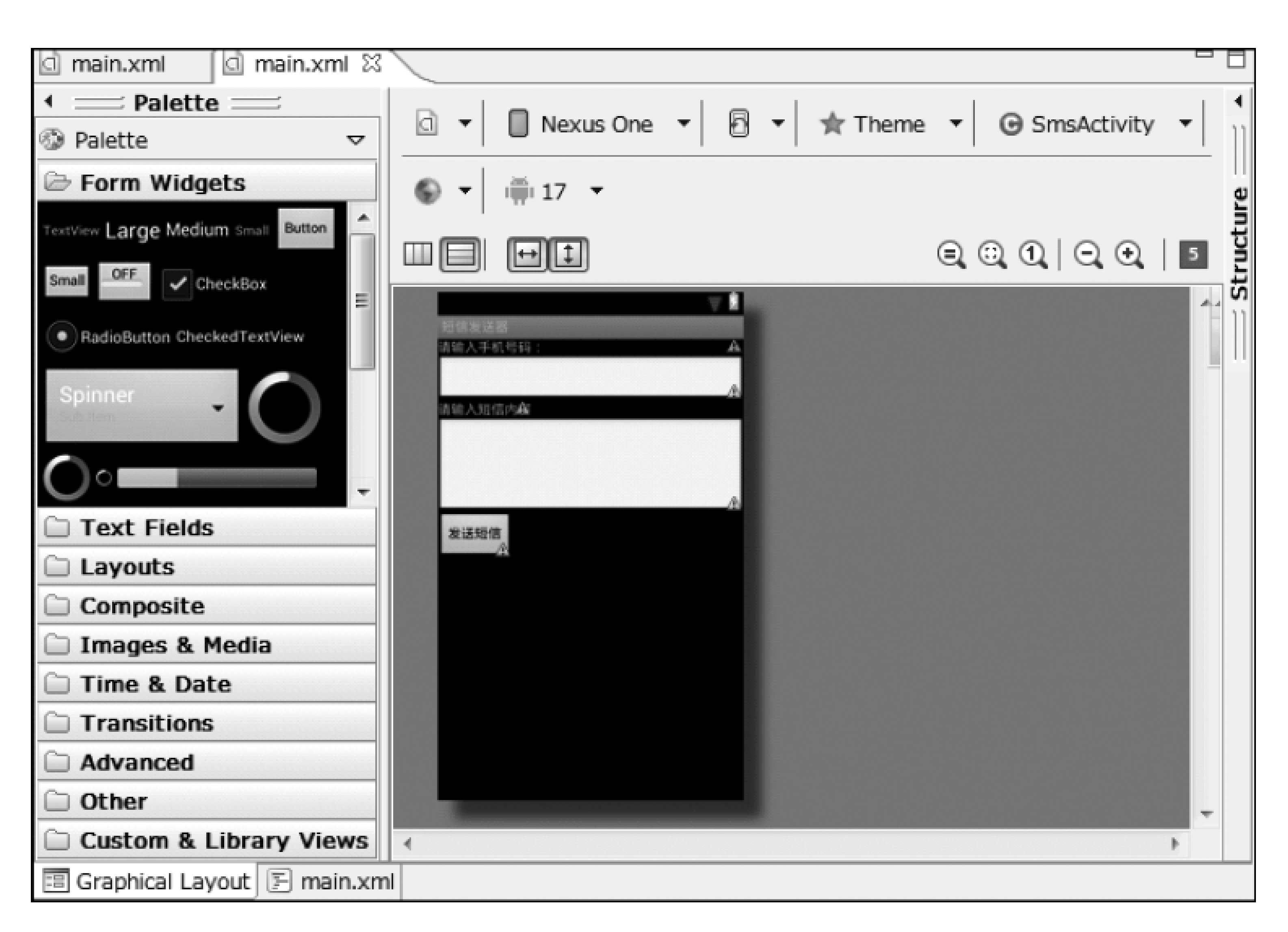


图 3-16　发送短信界面设计

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="请输入手机号码:" />

    <EditText
        android:id="@+id/et_mobile"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
    </EditText>

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="请输入短信内容" />

    <EditText
        android:id="@+id/et_content"
        android:layout_width="fill_parent"
        android:layout_height="100dp">
    </EditText>
```

```
    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="发送短信"
        android:onClick="sendsms" />
</LinearLayout>
```

3.3 项目功能的实现

代码编写实现的是通信应用程序的逻辑代码功能，该部分代码在项目的 src 目录中完成，如图 3-17 所示。

3.3.1 知识准备

1. Intent

Intent 提供了一种通用的消息系统，它允许在用户的应用程序与其他的应用程序间传递 Intent 执行动作和产生事件。使用 Intent 可以激活 Android 应用的三个核心组件：活动、服务和广播接收器。

一个 Intent 对象是对一次即将执行的操作的抽象描述，帮助我们在各个组件之间传递消息。一个 Intent 对象主要包含六类信息，但并非每个都需要包含这六类信息，如图 3-18 所示。

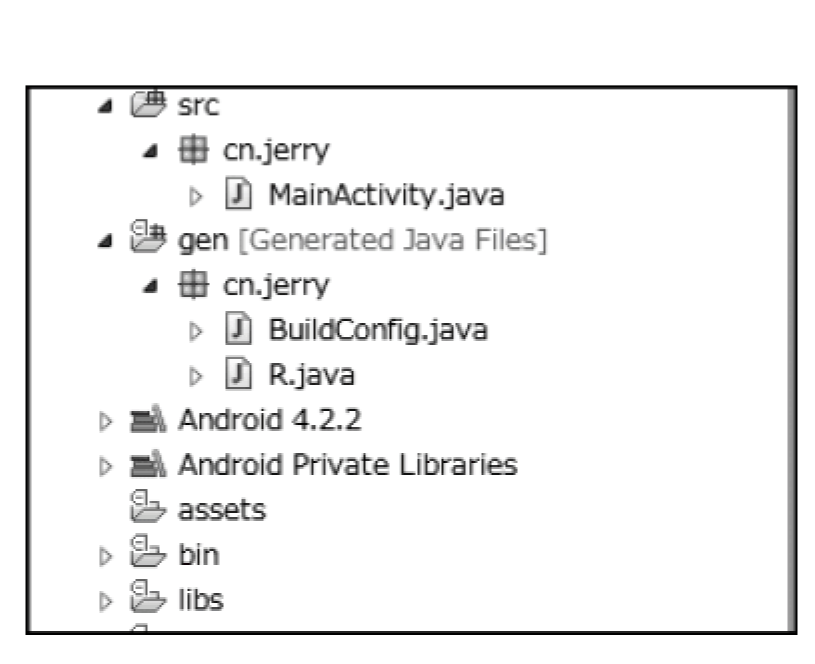


图 3-17 代码编写

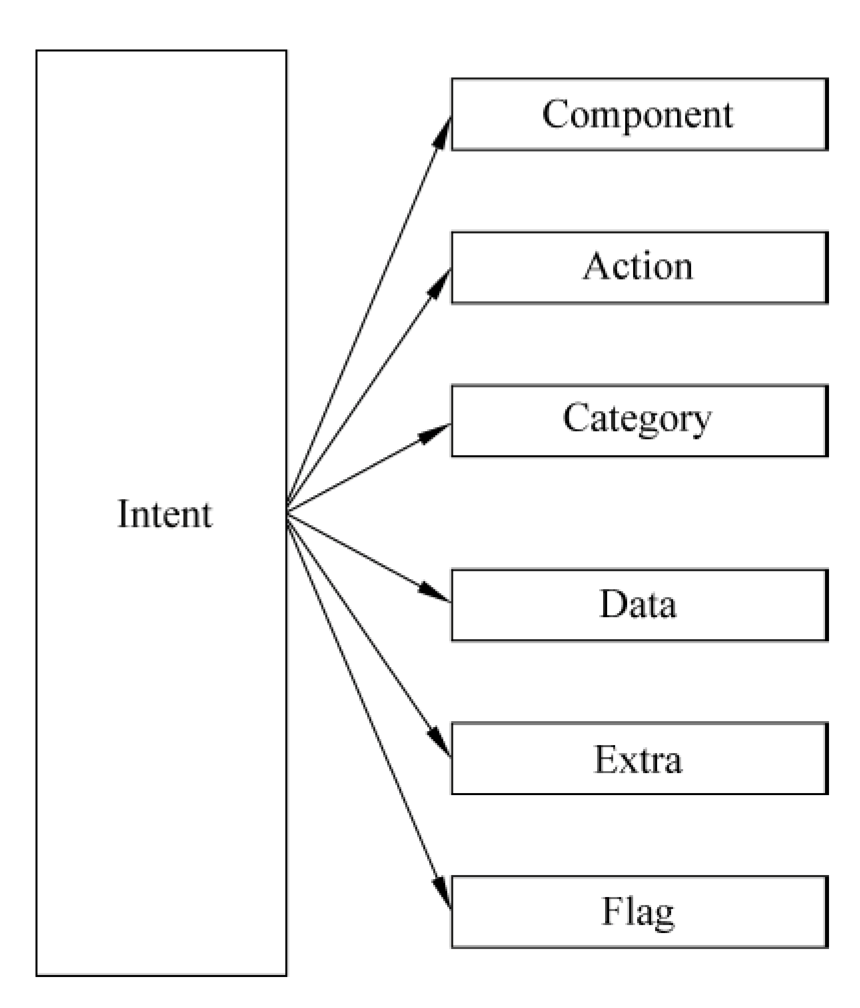


图 3-18 Intent 对象

(1) Intent 的 Component 属性

Intent 对象的 setComponent(ComponentName comp)方法用于设置 Intent 的 Component 属性。ComponentName 包含如下几个构造器。

- ComponentName(String pkg,String cls)。
- ComponentName(Context pkg,String cls)。
- ComponentName(Context pkg,Class<? >cls)。

由以上的构造器可知,创建一个 ComponentName 对象,需要指定包名和类名,这就可以唯一确定一个组件类,这样应用程序即可根据给定的组件类启动特定的组件。例如:

```
ComponentName comp=new ComponentName(FirstActivity.this, SecondActivity.class);
Intent intent=new Intent();
intent.setComponent(comp);
```

以上三句代码创建了一个 Intent 对象,并为其指定了 Component 属性,完全等价于下面的代码。

```
Intent intent=new Intent(FirstActivity.this, SecondActivity.class);
```

除了使用 setComponent()之外,还可以使用 setClass()、setClassName()显示指定目标组件,还可以调用 getComponent()方法获得 Intent 中封装的 ComponentName 对象。

当程序采用这种形式启动组件时,在 Intent 中明确地指定了待启动的组件类,此时的 Intent 属于显式 Intent,显式 Intent 应用场合比较狭窄,多用于启动本应用中的 Component,因为这种方式需要提前获知目标组件类的全限定名。而隐式 Intent 则通过 Intent 中的 action、category、data 属性指定目标组件需要满足的若干条件,系统筛选满足所有条件的 Component,从中选择最合适的 Component 或者由用户选择一个 Component 作为目标组件启动。

如果 Intent 中指定了 ComponentName 属性,Intent 的其他属性将被忽略。

(2) Intent 的 Action 属性

Action 属性是一个字符串,代表某一种特定的动作。Intent 类预定义了一些 Action 常量,开发者也可以自定义 Action。一般来说,自定义的 Action 应该以 Application 的包名作为前缀,然后附加特定的大写字符串,例如 cn. xing. upload. action. UPLOAD_COMPLETE 就是一个命名良好的 Action。

Intent 类的 setAction()方法用于设定 Action,getAction()方法可以获取 Intent 中封装的 Action。

以下是 Intent 类中预定义的部分 Action。

- ACTION_CALL 目标组件为 Activity,表示拨号动作。
- ACTION_EDIT 目标组件为 Activity,表示向用户显示数据以供其编辑的动作。
- ACTION_MAIN 目标组件为 Activity,表示作为 task 中的初始 Activity 启动。
- ACTION_BATTERY_LOW 目标组件为 broadcastReceiver,提醒手机电量过低。
- ACTION_SCREEN_ON 目标组件为 broadcast,表示开启屏幕。

(3) Intent 的 Category 属性

Category 属性也是一个字符串,用于指定一些目标组件需要满足的额外条件。Intent 对象中可以包含任意多个 Category 属性。Intent 类也预定义了一些 Category 常

量，开发者也可以自定义 Category 属性。

Intent 类的 addCategory()方法为 Intent 添加 Category 属性，getCategories()方法用于获取 Intent 中封装的所有 Category。

以下是 Intent 类中预定义的部分 Category。

- CATEGORY_HOME 表示目标 Activity 必须是一个显示 home screen 的 Activity。
- CATEGORY_LAUNCHER 表示目标 Activity 可以作为 task 栈中的初始 Activity，常与 ACTION_MAIN 配合使用。
- CATEGORY_GADGET 表示目标 Activity 可以被作为另一个 Activity 的一部分嵌入。

(4) Intent 的 Data 属性

Data 属性指定所操作数据的 URI。Data 经常与 Action 配合使用，如果 Action 为 ACTION_EDIT，Data 的值应该指明被编辑文档的 URI；如果 Action 为 ACTION_CALL，Data 的值应该是一个以 tel：开头并在其后附加号码的 URI；如果 Action 为 ACTION_VIEW，Data 的值应该是一个以 http：开头并在其后附加网址的 URI。

Intent 类的 setData()方法用于设置 Data 属性，setType()方法用于设置 Data 的 MIME 类型，setDataAndType()方法可以同时设定两者。可以通过 getData()方法获取 Data 属性的值，通过 getType()方法获取 Data 的 MIME 类型。

(5) Intent 的 Extra 属性

通过 Intent 启动一个 component 时，经常需要携带一些额外的数据过去。携带数据需要调用 Intent 的 putExtra()方法，该方法存在多个重载方法，可用于携带基本数据类型及其数组、String 类型及其数组、Serializable 类型及其数组、Parcelable 类型及其数组、Bundle 类型等。Serializable 和 Parcelable 类型表示一个可序列化的对象，Bundle 与 Map 类似，可用于存储键值对。

(6) Intent 的 Flag 属性

Flag 属性是一个 int 值，用于通知 Android 系统如何启动目标 Activity，或者启动目标 Activity 之后应该采取怎样的后续操作。所有的 Flag 都在 Intent 类中定义，部分常用 Flag 如下：

- FLAG_ACTIVITY_NEW_TASK 通知系统将目标 Activity 作为一个新 task 的初始 Activity。
- FLAG_ACTIVITY_NO_HISTORY 通知系统不要将目标 Activity 放入历史栈中。
- FLAG_FROM_BACKGROUND 通知系统这个 Intent 来源于后台操作，而非用户的直接选择。

(7) IntentFilter 类

IntentFilter 类表示 Intent 过滤器，大部分情况下，每一个 component 都会定义一个或多个 IntentFilter，用于表明其可处理的 Intent。

一般来说，Component 的 IntentFilter 应该在 AndroidManifest.xml 文件中定义。

定义的方法为在<Activity>、<receiver>、<service>元素中增加一个或多个<intent-filter>子元素。例如：

```
<!--声明作为程序入口的Activity-->
<Activity android:name=".FirstActivity">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</Activity>
```

（8）IntentFilter与隐式Intent

Android系统处理隐式Intent时，会比较Intent和IntentFilter的Action、Data、Category属性，如果以上3个属性全都相符，则IntentFilter所属的zomponent就可以作为目标组件的候选（存在多个符合条件的Component时）。

① 测试Action属性。Intent最多只能定义1个Action，而filter可以定义1个或多个Action。通过Action测试的条件为：filter定义了Intent的Action。例如Intent的Action为android.intent.action.MAIN，则定义了android.intent.action.MAIN的filter都能通过Action测试（filter还可以包含更多额外的Action）。

如果filter没有定义Action，则filter将阻塞所有Intent。如果Intent没有定义Action，那么只要filter定义了Action就可以通过Action测试。

② 测试Category属性。Intent可以有任意多个Category，filter也可以有任意个Category。通过Category测试的条件为filter定义了Intent的所有Category。例如Intent定义了android.intent.category.DEFAULT和cn.xing.intent.category.UPLOAD这两个Category，则定义了以上两个Category属性的filter才能通过测试（filter还可以包含更多额外的Category）。

根据上面的规则，如果一个Intent没有定义Category，则所有filter都可以通过Category测试。但是有一种例外，即以startActivity(intent)方式启动一个Activity时，系统会为Intent增加一个值为android.intent.category.DEFAULT的Category，这就意味着每一个期望通过Category测试的Activity，都要在其filter中定义android.intent.category.DEFAULT（除了作为程序入口的Activity外）。

③ 测试Data属性。Intent最多只能定义1个Data，filter则可以定义多个Data。

通过Data测试的条件如下。

a. 如果Intent没有指定Data和Data type，则只有没有定义Data和Date type的filter才能通过测试。

b. 如果Intent定义了Data而没有定义Data type，则只有定义了相同Data且没有定义Date type的filter才能通过测试。

c. 如果Intent没有定义Data却定义了Data type，则只有未定义Data且定义了相同的Data type的filter才能通过测试。

d. 如果Intent既定义了Data也定义了Dta type，则只有定义了相同的Data和Data type的filter才能通过测试。

Data 属性是一个 URI，URI 中包含 scheme、host、port 和 path，典型的 URI 为：

```
scheme://host:port/path
```

scheme、host、port 和 path 都是可选的。比较两个 Data 时，只比较 filter 中包含的部分。比如 filter 的一个 Data 只是指定了 scheme 部分，则测试时只是比较 Data 的 scheme 部分，只要两者的 scheme 部分相同，就视为“相同的 Data”。

(9) Intent 用法实例

① Activity 跳转。

```
//跳转到 secondActivity
//1.新建意图
Intent intent= new Intent();
//2.把第一个 Activity 和第二个 Activity 放入意图
intent.setClass(FirstActivity.this, SecondActivity.class);
//3.实现跳转
startActivity(intent);
//ps:2个 Activity 都要在 AndroidMenifest.xml 中进行注册,以下是 SecondActivity 的注册代码
< Activity
    android:label= "@ string/app_name"
    android:name= ".SecondActivity">
< /Activity>
```

② 向下一个 Activity 传递数据，使用 Bundle 和 Intent. putExtras 实现。

```
//1.获取用户名和密码
EditText editText= (EditText) findViewById(R.id.et_account);
EditText editText2= (EditText) findViewById(R.id.et_password);
String account= editText.getText().toString();
String pwd= editText2.getText().toString();
//2.新建 intent
Intent intent= new Intent();
//3.把参数放入 intent 中
intent.putExtra("account", account);
intent.putExtra("pwd", pwd);
intent.setClass(Login.this, Result.class);
//4.跳转
startActivity(intent);
对于数据的获取可以采用：
Intent intent= getIntent();
String account= intent.getStringExtra("account");
String pwd= intent.getStringExtra("pwd");
```

2. Activity

(1) 什么是 Activity

Activity 是用户接口程序，原则上它会提供给用户一个交互式的接口功能，是 Android 应用程序的基本功能单元。Activity 本身是没有界面的，所以 Activity 类创建了一个窗口，开发人员可以通过 setContentView(View)接口把 UI 放到 Activity 创建的窗口上，当 Activity 指向全屏窗口时，也可以用其他方式实现。作为漂浮窗口（通过 windowIsFloating 的主题集合），或者嵌入其他的 Activity(使用 ActivityGroup)。

(2) Activity 生命周期

Activity 生命周期如图 3-19 所示。在一个 Activity 正常启动过程中，这些方法调用的顺序是 onCreate→onStart→onResume；在 Activity 被 kill 时的顺序是 onPause→onStop→onDestroy，这是一个完整的生命周期。那么对于中断处理（比如电话来了），则是 onPause→onStop，恢复时 onStart→onResume；如果当前应用程序是一个 Theme 为 Translucent（半透明）或者 Dialog 的 Activity，那么中断就是 onPause，恢复时是 onResume。

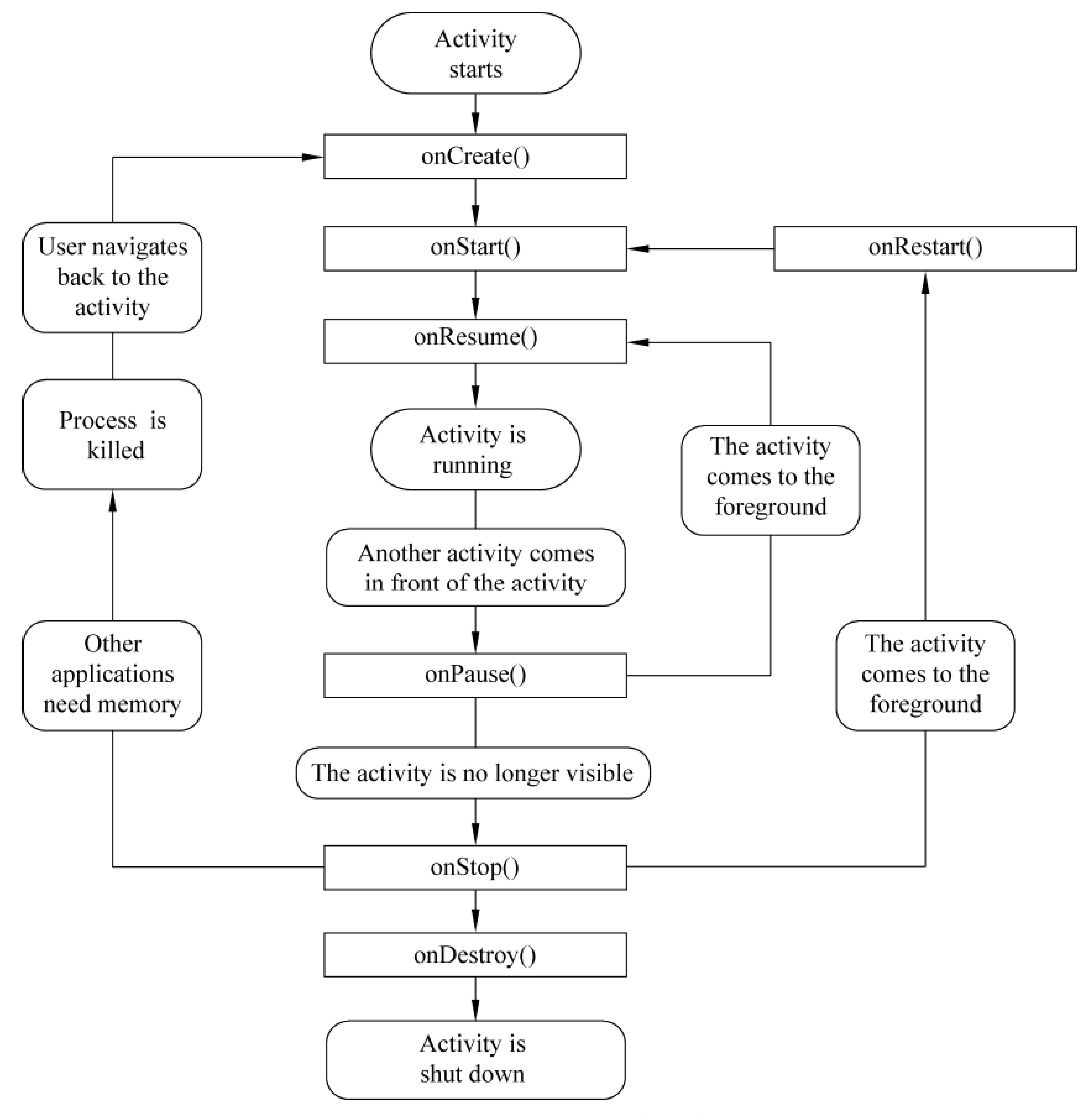


图 3-19 Activity 生命周期

对于 Other app need memory，就是手机在运行一个应用程序时，有可能打来电话、发来短信，或者电力不足了，这时候程序都会被中断，优先去服务电话的基本功能，另外系统也不允许用户占用太多资源，至少要保证一些功能（比如电话），所以资源不足时也就有可

能被中断。

上述 Activity 生命周期涉及的方法作用如下。

- onCreate：在这里创建界面，做一些数据的初始化工作。
- onStart：到这一步变成“用户可见不可交互”的状态。
- onResume：变成和用户可交互的状态(在 Activity 栈系统通过栈的方式管理这些 Activity，即当前 Activity 在栈的最上端，运行完弹出栈，则回到上一个 Activity)。
- onPause：到这一步是可见但不可交互的，系统会停止动画等消耗 CPU 的事情。从上文的描述已经知道，应该在这里保存一些数据，因为这个时候用户的程序的优先级降低，有可能被系统收回。在这里保存的数据，应该在 onResume 里读出来。
- onStop：变得不可见，被下一个 Activity 覆盖了。
- onDestroy：这是 Activity 被 kill 前最后一个被调用的方法，可能是其他类调用 finish 方法或者是系统为了节省空间将它暂时性的关闭，可以用 isFinishing()判断。如果用户有 一个 Progress Dialog 在线程中运行，请在 onDestroy 里将其取消，不然等线程结束时，调用 Dialog 的 cancel 方法会出现异常。
- onPause、onStop、onDestroy 三种状态下 Activity 都有可能被系统 kill。
- Activity 的使用需要在 Manifest 文件中添加相应的＜Activity＞，并设置其属性和 intent-filter。

(3) Activity 生命周期实例

当第二个 Activity 完全遮住第一个 Activity 时的生命周期如下。

firstActivity 执行过程：onCreate→onStart→onResume→onPause→onStop→onRestart→onStart→onResume。

```
/** Called when the Activity is first created. * /
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        System.out.println("firstActivity--->oncreate");
    }

    @Override
    protected void onStart() {
        //TODO Auto-generated method stub
        super.onStart();
        System.out.println("firstActivity--->onstart");
    }

    @Override
    protected void onResume() {
        //TODO Auto-generated method stub
        super.onResume();
        System.out.println("firstActivity--->onresume");
```

```
}

@Override
protected void onPause() {
    //TODO Auto-generated method stub
    super.onPause();
    System.out.println("firstActivity--->onpause");
}

@Override
protected void onStop() {
    //TODO Auto-generated method stub
    super.onStop();
    System.out.println("firstActivity--->onstop");
}

@Override
protected void onDestroy() {
    //TODO Auto-generated method stub
    super.onDestroy();
    System.out.println("firstActivity--->ondestory");
}

@Override
protected void onRestart() {
    //TODO Auto-generated method stub
    super.onRestart();
    System.out.println("firstActivity--->onresratr");
}
/**
 * 跳转到 SecondActivity
 * @param view
 */
public void jump(View view){
    Intent intent=new Intent();
    intent.setClass(FirstActivity.this, SecondActivity.class);
    startActivity(intent);
}
```

（4）Activity 实例 2

当第二个 Activity 没有完全遮住第一个 Activity 时的生命周期如下。

firstActivity 执行过程：onCreate→onStart→onResume→onPause→onResume。

```
<Activity
        android:label="@string/app_name"
        android:name=".SecondActivity" android:theme="@android:style/Theme.Dialog">
</Activity>
```

3．permission（允许）

（1）Android 安全机制概述

Android 是一个权限分离的系统。这是利用 Linux 已有的权限管理机制，通过为每一个 Application 分配不同的 uid 和 gid，从而使不同的 Application 之间的私有数据和访问（native 以及 java 层通过这种 sandbox 机制）达到隔离的目的。与此同时，Android 还在此基础上进行扩展，提供了 permission 机制，它主要是用来对 Application 可以执行的某些具体操作进行权限细分和访问控制，同时提供了 per-URI permission 机制，用于对某些特定的数据块进行 ad-hoc 方式的访问。

（2）permission

一个权限主要包含三个方面的信息：权限的名称；所属的权限组；保护级别。一个权限组是指把权限按照功能分成的不同的集合。每一个权限组包含若干具体权限，例如在 COST_MONEY 组中包含 android. permission. SEND_SMS、android. permission. CALL_PHONE 等和费用相关的权限。

每个权限通过 protectionLevel 标识保护级别：normal、dangerous、signature、signatureorsystem。不同的保护级别代表了程序要使用此权限时的认证方式。normal 的权限是只要申请了就可以使用；dangerous 的权限在安装时需要用户确认才可以使用；signature 和 signatureorsystem 的权限需要使用者的 APP 和系统使用同一个数字证书。

Package 的权限信息主要通过在 AndroidManifest. xml 中通过一些标签指定。如 <permission>标签、< permission-group >标签、< permission-tree >标签等。如果 Package 需要申请使用某个权限，那么需要使用 <use-permission>标签指定。

程序执行需要读取到安全敏感项，则必须在 androidmanifest. xml 中声明相关权限请求。

3.3.2　项目功能相关代码设计

（1）拨打电话逻辑代码

```
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class PhoneActivity extends Activity {
    /* * Called when the Activity is first created. * /
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void call(View view){
        //实现打电话功能
        //1.拿到电话号码
        EditText editText= (EditText) findViewById(R.id.et_mobile);
```

```
            String telPhone= editText.getText().toString();
            Intent intent= new Intent(Intent.ACTION_CALL,Uri.parse("tel:"+
            telPhone));
            //2.启动拨打电话
            startActivity(intent);
        }
    }
//在 androidmenifest.xml 中加入打电话的权限
<uses-permission android:name="android.permission.CALL_PHONE"/>
```

（2）发送短信逻辑代码

```
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.telephony.TelephonyManager;
import android.view.View;
import android.widget.EditText;

public class SmsActivity extends Activity {
    /** Called when the Activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    /**
     * 发送短信
     * @param view
     */
    public void sendsms(View view){
        //1.拿到电话号码和内容
        String mobile= ((EditText)findViewById(R.id.et_mobile)).getText().toString();
        String content= ((EditText)findViewById(R.id.et_content)).getText().toString();
        TelephonyManager tm= (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);
        String deviceid= tm.getDeviceId();
        String tel= tm.getLine1Number();

        //2.新建短信的 Intent
        Intent intent= new Intent(Intent.ACTION_SENDTO, Uri.parse("smsto:"+ tel));
        //3.把短信内容放入 intent
        intent.putExtra("sms_body", content);
        //4.发送短信
        startActivity(intent);
    }
}
//在 AndroidMenifest.xml 文件中要加入发送短信的权限
<uses-permission android:name="android.permission.SEND_SMS"/>
```

(3) 发送短信代码优化

发送短信有字数的限制，目前每单位短信最多是 140 个英文字符或 70 个汉字符，超过这个限制，短信将自动分割成相应条数(按条数收费)，并在收件人的手机上自动组合。Android 系统中使用 SmsManager 管理每条短信要显示的字数。

```
SmsManager smsManager= SmsManager.getDefault();
    //分割短信内容
    ArrayList<String> texts= smsManager.divideMessage(content);
    for (String text : texts) {
        smsManager.sendTextMessage(tel, null, text, null, null);
    }
```

3.4　系统运行与效果测试

系统运行与效果测试如图 3-20～图 3-25 所示。

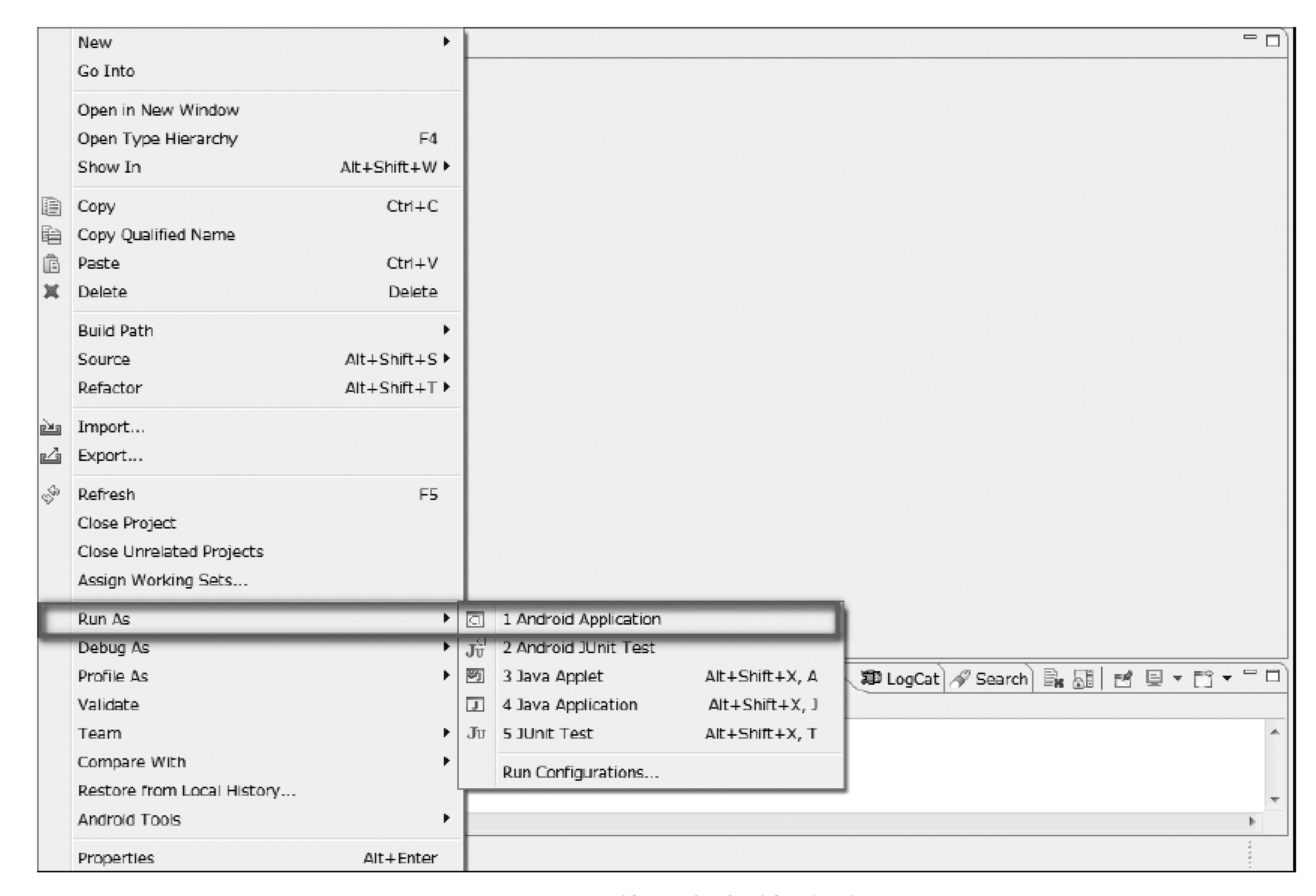


图 3-20　系统运行与效果测试



图 3-21 输入电话号码



图 3-22 电话呼叫

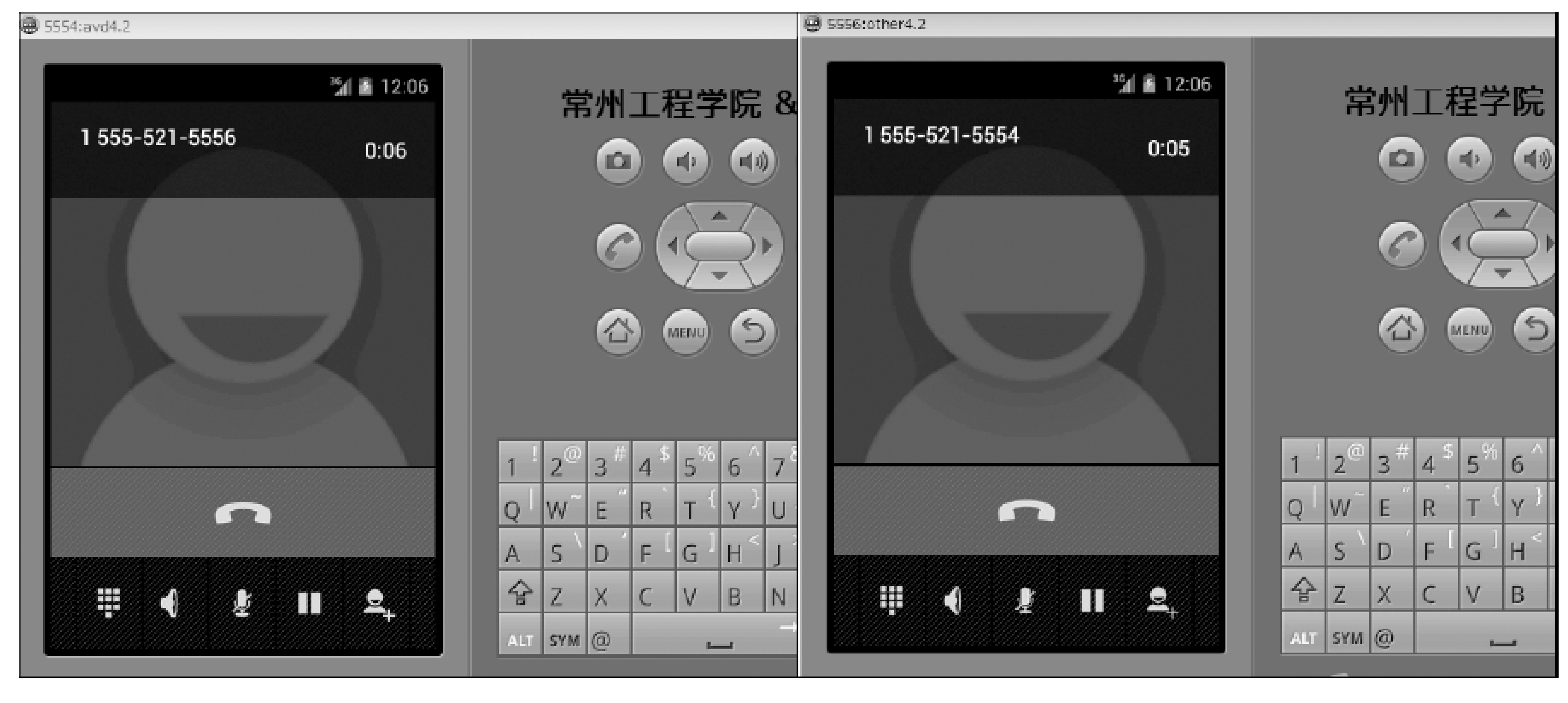


图 3-23 通话



图 3-24　输入短信

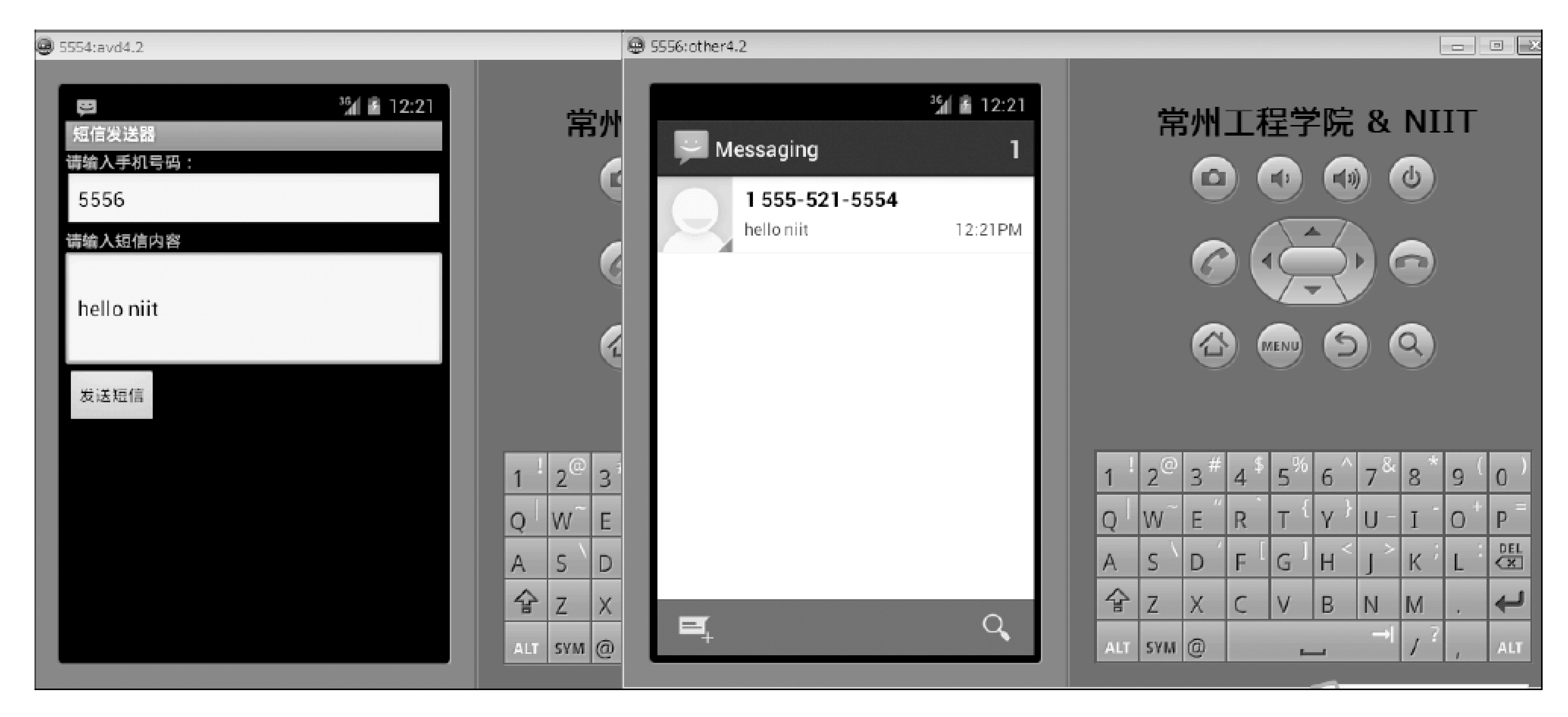


图 3-25　发送短信

3.5　本 章 小 结

通过本章项目练习，学习了 Android 的三个基本控件 TextView、EditText、Button，Android 四大组件之一 Activity，消息传递机制 Intent 和 Android 权限控制 permission。

3.6 项目实践

（1）试着把做好的APP安装到实验手机中，测试拨打电话和发送短信的功能并分析结果。

（2）使用不同分辨率的模拟器测试效果。

（3）使用不同版本的模拟器测试效果。

（4）使用Intent和Activity实现登录功能。

（5）理解Android中的权限设计机制。

（6）编写代码实现软件安装和卸载。

（7）编写代码访问浏览器。

第 4 章 水果连连看的设计及开发

本章的工作目标如下：

(1) 使用对话框实现菜单和选关界面。

(2) 自定义 View 实现游戏布局。

(3) 实现连连看算法。

4.1 项目分析

水果连连看是在一堆图案中找出相同图案进行配对的简单游戏，鼠标分别单击相同的两个水果图片，如果连接这两个水果图片的直线不超过三条，并且不碰到其他水果图片，则消除这两个水果图片，直到最终将图片全部消除，游戏有时间限制和“提示”功能。

首先介绍一下 Android 中连连看项目的架构，对所用到的技术进行简要分析，项目架构如图 4-1 所示。

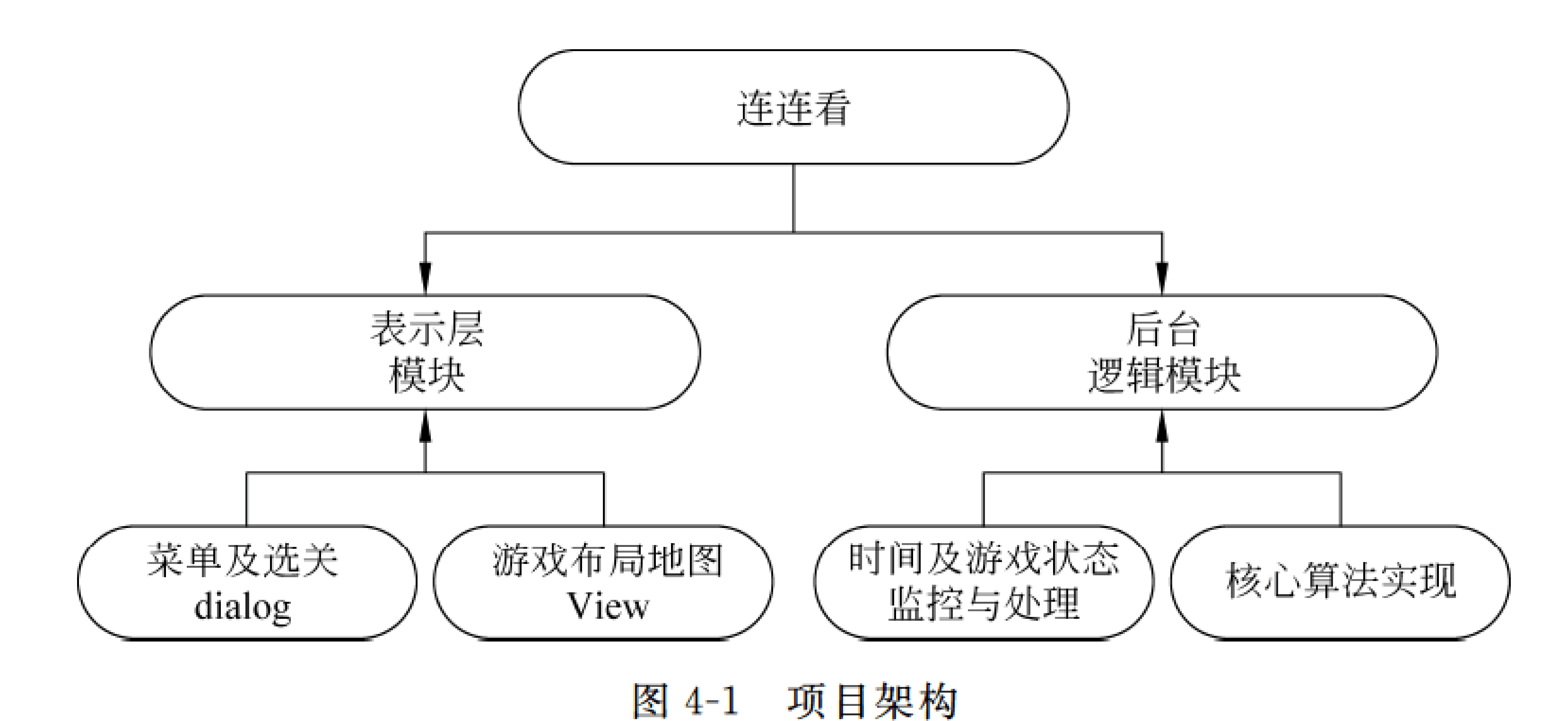


图 4-1　项目架构

本程序主要包含两大模块：表示层模块和后台逻辑模块。

其中表示层模块可以理解为游戏的 UI 及一些辅助效果，表示层模块中，重要的是实现游戏的布局地图。项目中，游戏的布局将使用自定义 View 的方式，在屏幕上贴图实现。而菜单模块及选关的 dialog，只是为用户提供一些常见的选择，如重玩、过关继续等，为了有一个更好的用户交互环境，dialog 将通过自定义 dialog 的方式实现。

而后台逻辑模块中，对于程序计算的实现与程序各种状态的监听，将是整个程序运行

的基础。此模块中将实现对于游戏剩余时间限制和游戏状态的监听与处理。对于游戏剩余时间的限制,将开启单独的线程进行处理,从而不至于影响主程序逻辑的运行;游戏的状态的监听处理中,将会实现对于连通的两个图标的消除(即游戏界面的更新)、游戏输赢的监听判断、游戏暂停与否等(暂停状态需要同时将剩余时间暂停,而时间监听线程需要知道所处状态,二者紧密联系)。

4.2 连连看算法

本程序中最重要的是核心算法模块的实现,在游戏中,最主要的算法是判断两个选中的图标是否能够连通,其余两个算法也依赖于此算法。

在介绍连接算法之前,先以 4×4 的棋盘为例,简单介绍一下连连看的布局算法。

在程序初始化时,先将要加载的图片在棋盘上按序绘制出来,注意每一种图标在绘制时需要一次性绘制两次,这样,才能保重绘制出来的每种图标都是偶数个。假设最初如图 4-2 所示。

这样绘制后,随机地调换棋盘中的图标(是现有棋盘中的图标之间的调换,并不是更改成为其他的图标)。调换后的棋盘可能如图 4-3 所示,这样就完成了棋盘的初始化。实际上,棋盘在最外面一层是不添加图标的,此处为的是连线时能够使用最外层画线,而不会出现穿过图标画线的情况,棋盘如图 4-4 所示。

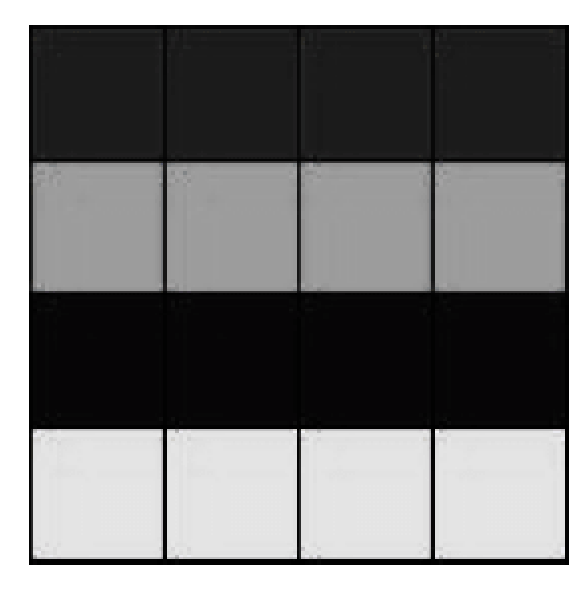


图 4-2 最初绘制的棋盘

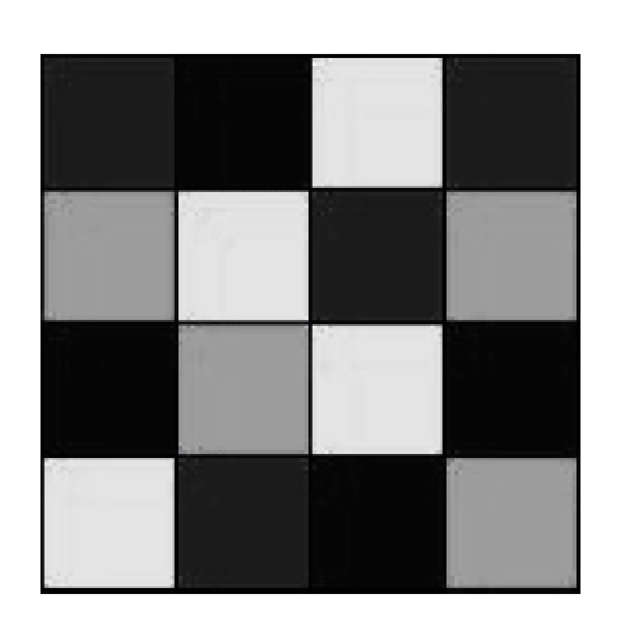


图 4-3 调换后的棋盘

连连看算法中两个图标能够连接的充分条件是:①两个图标是相同的;②两个图标之间有一条路相连,其中这条“路上”没有其他的图标“阻碍”;③这一条路不能有两个以上的拐角。同时满足这三个条件即可认为两个图标是相连通的。对于连通的判断中,图标连通时有以下三种情况。

(1) 直线型:直线型是从横向或纵向单方向判断的情况,只要两者之间没有其他图标即可连通,这种情况最容易判断。

(2) 一折型:一折型是在两个选中图标确定的两个对角顶点画一个矩形,若是其余两个顶点中有满足与两个选中图标都能够“直线型”相连的,即可认为这两个选中图标可以连通,如图 4-5 所示。

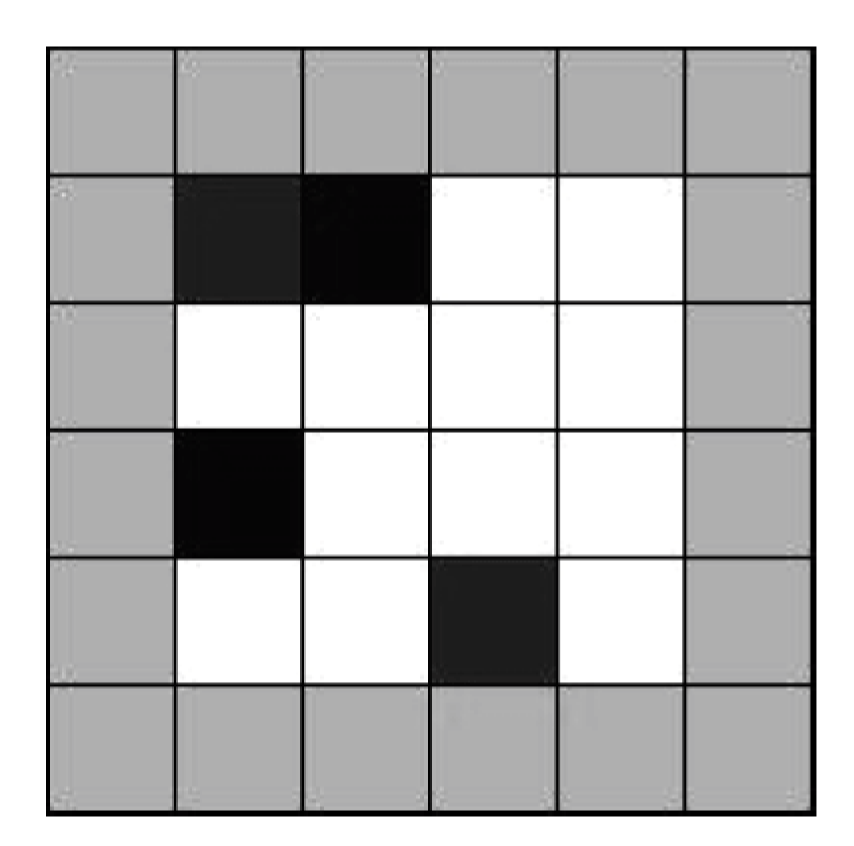


图 4-4　棋盘及最外层

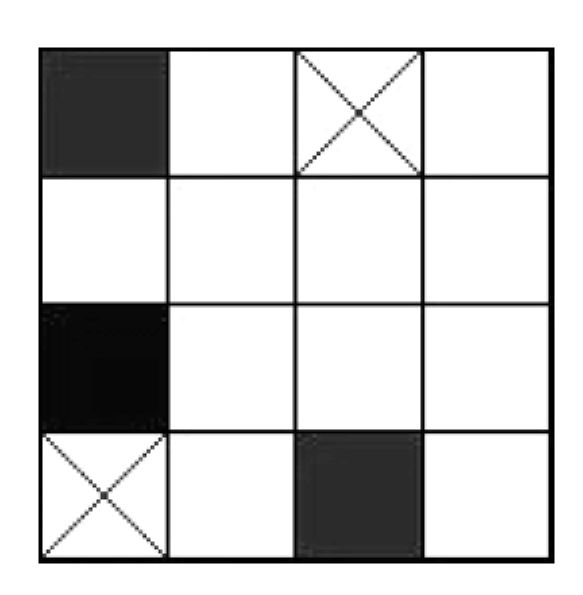


图 4-5　一折型的示例

(3) 二折型：对于二折型的判断是重点。判断二折型主要是做两个方向的扫描，即横向扫描与纵向扫描。

横向扫描中，首先将两个需要判断的图标(见图 4-6)进行横向扩展，扩展规则是在没有遇到其他图标时一直扩展，直到遇到此行的其他图标或者到达棋盘的边缘，扩展后的点如图 4-7 中⊠所示，如果扩展后的点中存在两点能够满足直线型连通的情况，如图 4-8 所示，即可判断这两个图标是可以连通的，连通的画线也是根据这两个辅助点相连而成的。

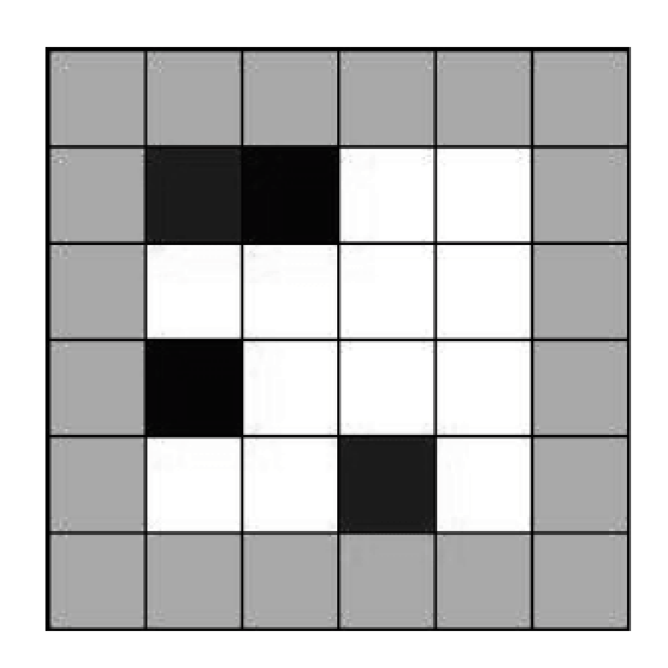


图 4-6　需要判断的两个灰色图标

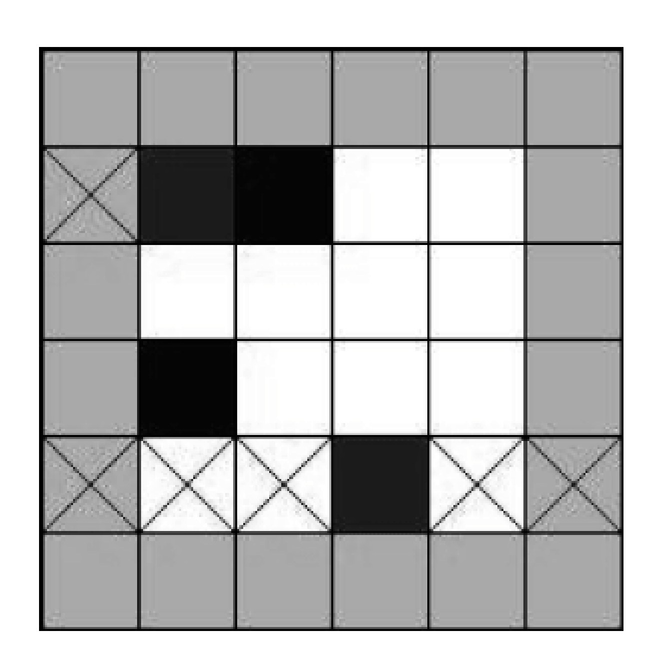


图 4-7　图标的横向扩展

纵向扫描与横向扫描类似，如图 4-9 和图 4-10 所示。

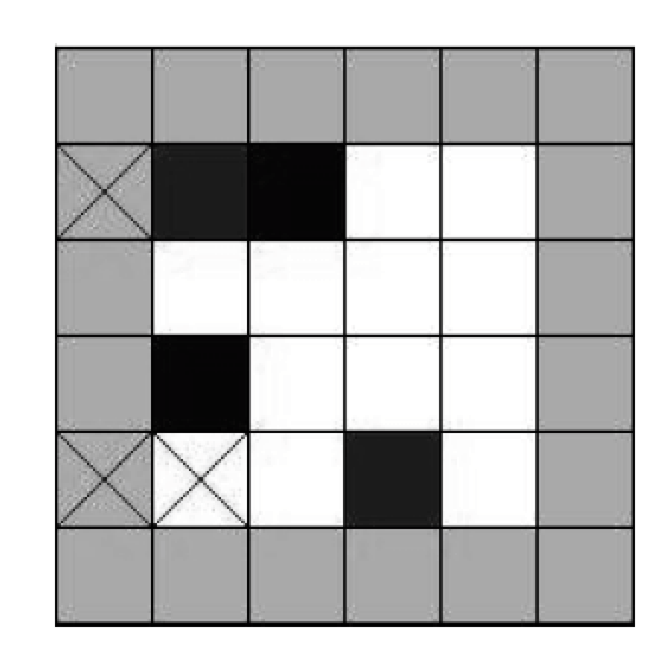


图 4-8　扩展中存在两点能"直线型"连通 1

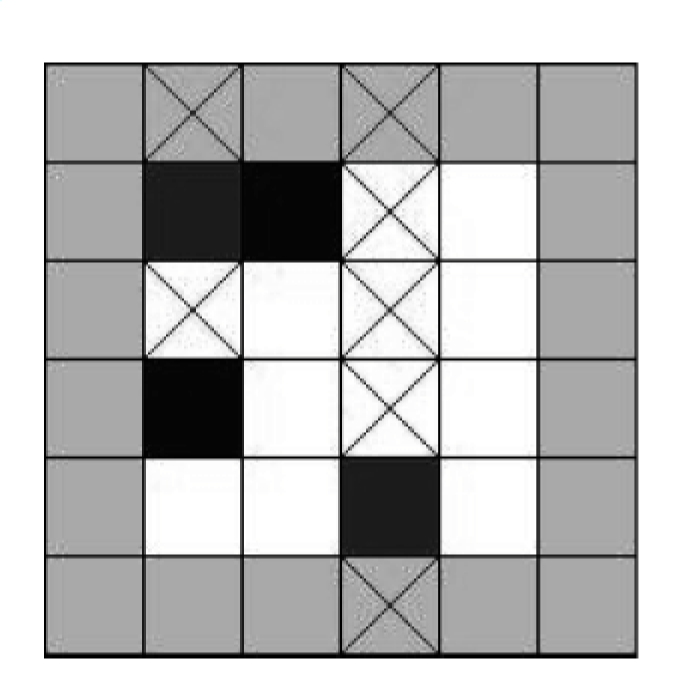


图 4-9　图标的纵向扩展

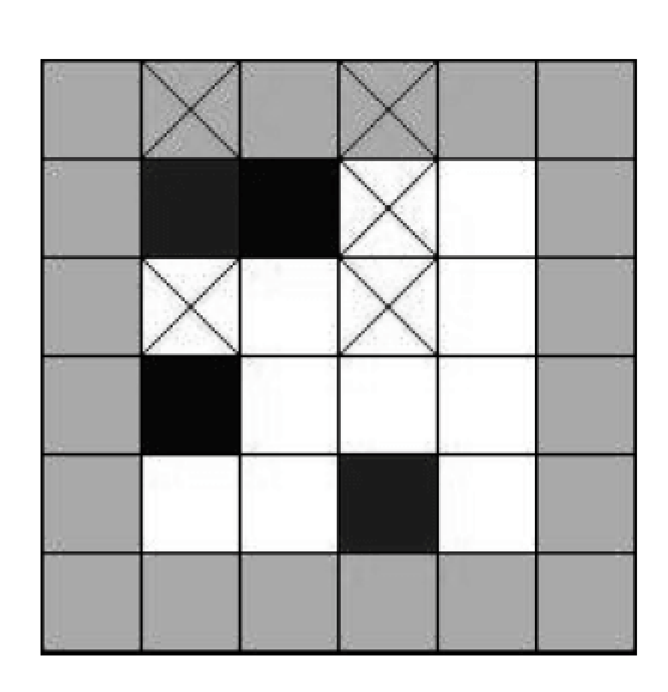


图 4-10 扩展中存在两点能“直线型”连通 2

4.3 项目界面设计

水果连连看的界面设计主要包括以下几个方面：①自定义游戏界面；②使用滑块产生倒计时效果；③使用图片控件和动画产生抖动效果；④使用渐变、描边、圆角的自定义背景。

4.3.1 知识准备

1. 图片控件

(1) ImageView

android.widget.ImageView 图片控件，继承自 android.view.View，在 android.widget 包中。最简单的使用方法是通过 src 属性设置图片路径，可引用 drawable 的图片。

XML 代码如下：

```
< ImageView android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:src="@drawable/a"/>
```

动态声明 ImageView，设置 ImageDrawable。

Java 代码如下：

```
ImageView iv2=(ImageView) findViewById(R.id.imageView2);
iv2.setImageDrawable(getResources().getDrawable(R.drawable.b));
```

效果如图 4-11 所示。

(2) ImageButton

android.widget.ImageButton 图片控件，继承自 android.widget.ImageView，在 android.widget 包中。最简单的使用方法是通过 src 设置图片路径，可引用 drawable 的图片。

XML 代码如下：

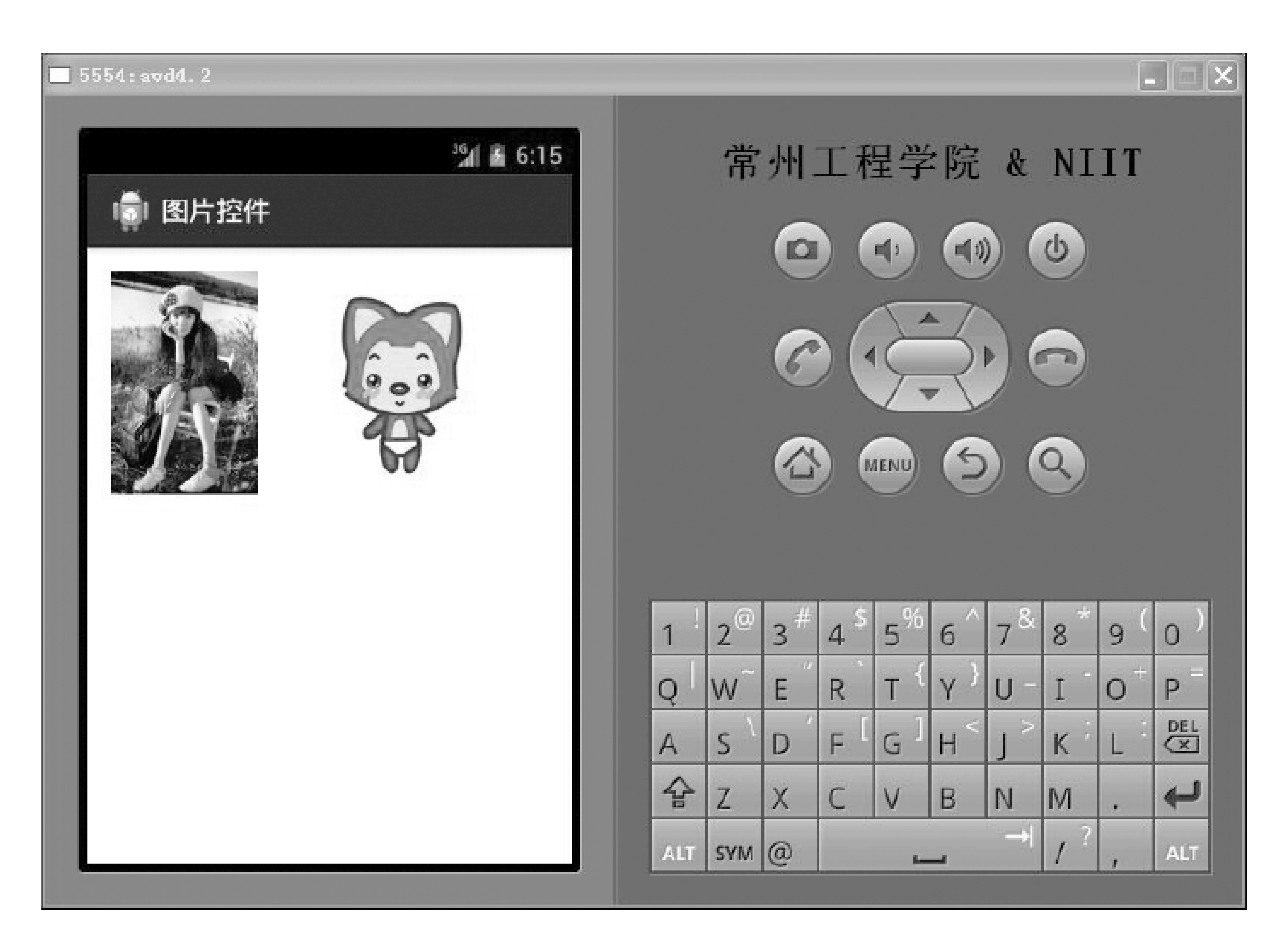


图 4-11　图片控件

```
<ImageButton android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:src="@drawable/but_01"/>
```

动态声明 ImageButton，设置 ImageDrawable。

Java 代码如下：

```
ImageButton ib= (ImageButton) findViewById(R.id.imageButton2);
ib.setImageDrawable(getResources().getDrawable(R.drawable.b2));
```

效果如图 4-12 所示。



图 4-12　图片按钮控件

2. 滑块控件

SeekBar 可以作为音乐播放器的进度指示和调整工具、音量调整工具等，SeekBar 是 ProgressBar 的一个子类，下面用一个可以改变并显示当前进度的拖动条例子演示它的使用。

XML 代码如下：

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"  android:layout_width="fill_parent"
    android:layout_height="fill_parent">
   <SeekBar android:id="@+id/SeekBar01"  android:layout_width="245px"
       android:layout_height="25px"  android:paddingLeft="16px"
       android:paddingRight="15px"  android:paddingTop="5px"
       android:paddingBottom="5px"  android:progress="0"  android:max="0"
       android:secondaryProgress="0"  />
   <TextView android:layout_width="fill_parent"
       android:layout_height="wrap_content"  android:text="@string/hello"
       android:id="@+id/TextView01"  />
</LinearLayout>
```

Java 代码如下：

```
//找到拖动条和文本框
final SeekBar sb= (SeekBar) findViewById(R.id.SeekBar01);
final TextView tv1= (TextView) findViewById(R.id.TextView01);
//设置拖动条的初始值和文本框的初始值
sb.setMax(100);
sb.setProgress(30);
tv1.setText("当前进度：" + sb.getProgress());
//设置拖动条改变监听器
OnSeekBarChangeListener osbcl= new  OnSeekBarChangeListener() {

   @Override
   public void onProgressChanged(SeekBar seekBar, int progress,
          boolean fromUser) {
      tv1.setText("当前进度：" + sb.getProgress());
      Toast.makeText(getApplicationContext(), "onProgressChanged",
             Toast.LENGTH_SHORT).show();
   }

   @Override
   public void onStartTrackingTouch(SeekBar seekBar) {
      Toast.makeText(getApplicationContext(), "onStartTrackingTouch",
             Toast.LENGTH_SHORT).show();
   }
```

```
        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
            Toast.makeText(getApplicationContext(), "onStopTrackingTouch",
                    Toast.LENGTH_SHORT).show();
        }

    };

    //为拖动条绑定监听器
    sb.setOnSeekBarChangeListener(osbcl);
```

效果如图 4-13 所示。



图 4-13　滑块控件

3. 自定义 View

很多时候系统自带的 View 满足不了设计的要求，就需要自定义 View 控件。自定义 View 首先要实现一个继承自 View 的类。添加类的构造方法、override 父类的方法，如 onDraw、onMeasure 等。

具体步骤：首先新建一个 Android 工程，命名为自定义 View；然后自定义一个 View 类，命名为 MyView(extends View)。

Java 代码如下：

```
public class MyView extends View {
    private Paint mPaint;
    private Context mContext;
    private static final String mString= "Welcome to Mr Wei's blog";
```

```
    public MyView(Context context) {
        super(context);

    }
    public MyView(Context context,AttributeSet attr)
    {
        super(context,attr);

    }
    @Override
    protected void onDraw(Canvas canvas) {
        //TODO Auto-generated method stub
        super.onDraw(canvas);

        mPaint=new Paint();

        //设置画笔颜色
        mPaint.setColor(Color.RED);
        //设置填充
        mPaint.setStyle(Style.FILL);

        //画一个矩形,前两个是矩形左上角坐标,后两个是右下角坐标
        canvas.drawRect(new Rect(10, 10, 100, 100), mPaint);

        mPaint.setColor(Color.BLUE);
        //绘制文字
        canvas.drawText(mString, 10, 110, mPaint);
    }
}
```

然后将自定义的 View 加入 main.xml 布局文件,代码如下:

```
<cn.chap4.view.MyView
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
/>
```

效果如图 4-14 所示。

4. 动画

Android 平台提供了两类动画:一类是 Tween 动画,即通过对场景里的对象不断做图像变换(平移、缩放、旋转)产生动画效果;另一类是 Frame 动画,即顺序播放事先做好的图像,与电影类似。

Tween 动画通过对 View 的内容完成一系列的图形变换(包括平移、缩放、旋转、改变透明度)实现动画效果。具体来讲,Tween 动画需预先定义一组指令,这些指令指定了图形变换的类型、触发时间、持续时间。这些指令可以 XML 文件方式定义,也可以源代码方式定义。程序沿着时间线执行,这些指令就可以实现动画效果。动画的进度使用



图 4-14　自定义 View

Interpolator 控制，android 提供了几个 Interpolator 子类，实现了不同的速度曲线，如 LinearInterpolator 实现了匀速效果、Accelerateinterpolator 实现了加速效果、DecelerateInterpolator 实现了减速效果等。还可以定义自己的 Interpolator 子类，实现抛物线、自由落体等物理效果。

动画的运行模式有两种：①独占模式，即程序主线程进入一个循环，根据动画指令不断刷新屏幕，直到动画结束；②中断模式，即有单独一个线程对时间计数，每隔一定的时间向主线程发通知，主线程接到通知后更新屏幕。

图形变换通过仿射矩阵实现，是图形学中的基本知识，每种变换都是一次矩阵运算。Canvas 类中包含当前矩阵，当调用 drawBitmap(bmp，x，y，Paint)绘制时，Android 会先把 bmp 做一次矩阵运算，然后将运算的结果显示在 Canvas 上。这样，编程人员只需不断修改 Canvas 的矩阵并刷新屏幕，View 中的对象就会不停地做图形变换，动画也就形成了。

Android 中提供了 Animation、Interpolator、Transformation 等类具体实现 Tween 动画，下面逐一进行分析。

(1) Animation 类及其子类是动画的核心模块，它实现了各种动画效果，如平移、缩放、旋转、改变透明度等。

(2) Tween 动画的每一帧都根据 Interpolator 对 view 的内容做一次图形变换，因此 Animation 的核心工作是做变换(Transformation)。

(3) Aniamtion 是基类，它记录了动画的通用属性和方法。主要的属性包括动画持续时间、重复次数、Interpolator 等。动画中最重要的方法是 getTransformation(currentTime，outTransformation)，该方法根据当前间(currentTime)和 interpolator，计算当前的变换，在 outTransformation 中返回。

(4) TranslateAnimation、RotateAnimation、AlphaAnimation 等是 Animation 的子类,分别实现了平移、旋转、改变 Alpha 值等动画。

(5) 每个动画都重载了父类的 applyTransformation 方法,这个方法会被父类的 getTransformation 方法调用。另外,每个动画还有 initialize 方法,完成初始化工作。

(6) 不同的动画具有不同的属性,如 RotateAnimation 的属性是起始角度、终止角度和旋转点坐标,TranslateAnimation 的属性是起始位置和终止位置。AlphaAnimation 的属性是起始 Alpha 值和终止 Alpha 值。

Animation 类及其子类如图 4-15 所示。

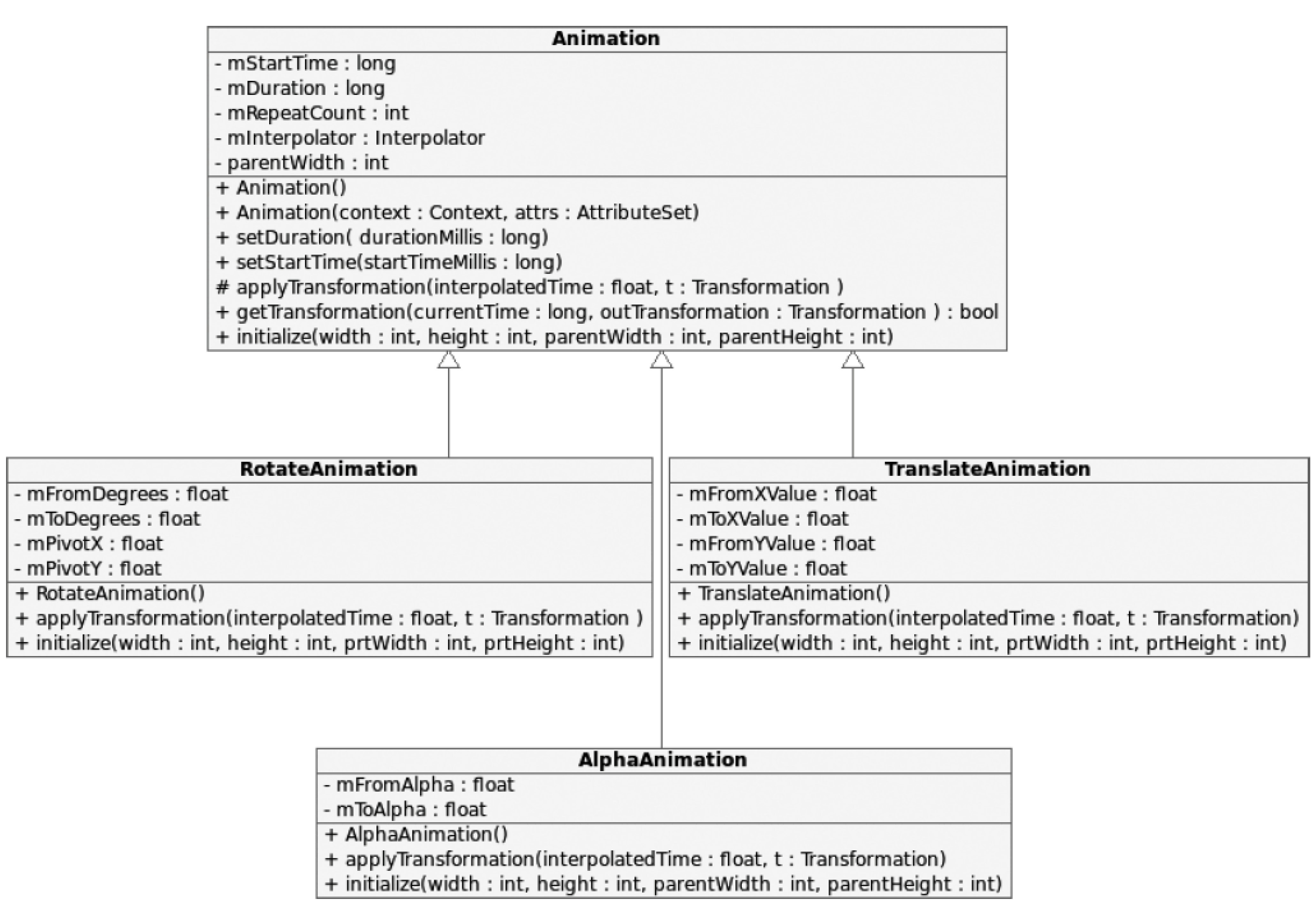


图 4-15 Animation 类及其子类

(7) Android 的 Animation 由四种类型组成

XML 代码中:①Alpha,渐变透明度动画效果;②Scale,渐变尺寸伸缩动画效果;③Translate,画面转换位置移动动画效果;④Rotate,画面转移旋转动画效果。

Java 代码中:①AlphaAnimation,渐变透明度动画效果;②ScaleAnimation,渐变尺寸伸缩动画效果;③TranslateAnimation,画面转换位置移动动画效果;④RotateAnimation,画面转移旋转动画效果。

在 XML 文件中定义动画,步骤如下:

① 打开 Eclipse,新建 Android 工程。

② 在 res 目录中新建 anim 文件夹。

③ 在 anim 目录中新建一个 myanim.xml(注意文件名小写)。

④ 加入 XML 的动画代码。

代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <alpha/>
    <scale/>
    <translate/>
    <rotate/>
</set>
```

Android 动画解析——XML 代码：

<alpha>

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
<alpha
android:fromAlpha="0.1"
android:toAlpha="1.0"
android:duration="3000"
/>
<!--透明度控制动画效果 alpha
    浮点型值：
        fromAlpha 属性为动画起始时透明度
        toAlpha   属性为动画结束时透明度
        说明：
            0.0表示完全透明
            1.0表示完全不透明
        以上值取 0.0~1.0的 float 数据类型的数字

    长整型值：
        duration属性为动画持续时间
        说明：
            时间以毫秒为单位
-->
</set>
```

<scale>

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <scale
        android:interpolator=
                "@android:anim/accelerate_decelerate_interpolator"
        android:fromXScale="0.0"
        android:toXScale="1.4"
        android:fromYScale="0.0"
        android:toYScale="1.4"
        android:pivotX="50%"
        android:pivotY="50%"
        android:fillAfter="false"
```

```
        android:duration="700" />
</set>
<!--尺寸伸缩动画效果 scale
        属性:interpolator 指定一个动画的插入器
        在我试验过程中,使用 android.res.anim 中的资源时发现
        有三种动画插入器:
            accelerate_decelerate_interpolator  加速-减速 动画插入器
            accelerate_interpolator             加速-动画插入器
            decelerate_interpolator             减速-动画插入器
        其他的属于特定的动画效果
      浮点型值:

            fromXScale 属性为动画起始时 X坐标上的伸缩尺寸
            toXScale   属性为动画结束时 X坐标上的伸缩尺寸

            fromYScale 属性为动画起始时 Y坐标上的伸缩尺寸
            toYScale   属性为动画结束时 Y坐标上的伸缩尺寸

            说明:
                以上四种属性值

                    0.0表示收缩到没有
                    1.0表示正常无伸缩
                    值小于 1.0表示收缩
                    值大于 1.0表示放大

            pivotX     属性为动画相对于物件的 X坐标的开始位置
            pivotY     属性为动画相对于物件的 Y坐标的开始位置

            说明:
                    以上两个属性值 从 0~100% 中取值
                    50% 为物件的 X或 Y方向坐标上的中点位置

        长整型值:
            duration属性为动画持续时间
            说明:时间以毫秒为单位

        布尔型值:
            fillAfter属性当设置为 true,该动画转化在动画结束后被应用
-->
```

<translate>

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
<translate
android:fromXDelta="30"
android:toXDelta="-80"
android:fromYDelta="30"
android:toYDelta="300"
```

```
android:duration="2000"
/>
<!--translate 位置转移动画效果
    整型值:
        fromXDelta 属性为动画起始时 X坐标上的位置
        toXDelta   属性为动画结束时 X坐标上的位置
        fromYDelta 属性为动画起始时 Y坐标上的位置
        toYDelta   属性为动画结束时 Y坐标上的位置
        注意:
            没有指定 fromXType toXType fromYType toYType 时,
            默认是以自己为相对参照物
    长整型值:
        duration 属性为动画持续时间
        说明:时间以毫秒为单位
-->
</set>
```

<rotate>

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
<rotate
    android:interpolator="@android:anim/accelerate_decelerate_
        interpolator"
    android:fromDegrees="0"
    android:toDegrees="+350"
    android:pivotX="50%"
    android:pivotY="50%"
    android:duration="3000" />
<!--rotate 旋转动画效果
    属性:interpolator 指定一个动画的插入器
        在我试验过程中,使用 android.res.anim 中的资源时发现
        有三种动画插入器:
            accelerate_decelerate_interpolator   加速-减速 动画插入器
            accelerate_interpolator              加速-动画插入器
            decelerate_interpolator              减速-动画插入器
        其他的属于特定的动画效果

    浮点数型值:
        fromDegrees 属性为动画起始时物件的角度
        toDegrees   属性为动画结束时物件旋转的角度,可以大于 360 度


        说明:
                当角度为负数,表示逆时针旋转
                当角度为正数,表示顺时针旋转
                (负数 from—to 正数:顺时针旋转)
                (负数 from—to 负数:逆时针旋转)
                (正数 from—to 正数:顺时针旋转)
                (正数 from—to 负数:逆时针旋转)
```

```
        pivotX      属性为动画相对于物件的 X 坐标的开始位置
        pivotY      属性为动画相对于物件的 Y 坐标的开始位置

        说明：      以上两个属性值 从 0~100% 中取值
                    50% 为物件的 X 或 Y 方向坐标上的中点位置

    长整型值：
        duration    属性为动画持续时间
        说明：      时间以毫秒为单位
-->
</set>
```

动画效果如图 4-16 所示。



图 4-16 动画

5. 渐变、圆角等效果

在 Android 开发过程中，经常需要改变控件的默认样式，那么通常会使用多个图片来解决。比如一个按钮，需要单击时的式样图片，默认的式样图片，这样就容易使 apk 变大。

除了使用 drawable 的图片外，还可以使用自定义图形 shape，Android 上支持 shape、gradient、stroke、corners、padding、solid 等属性。

XML 代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">

    <!--圆角-->
    <corners
```

```
        android:radius="9dp"
        android:topLeftRadius="2dp"
        android:topRightRadius="2dp"
        android:bottomLeftRadius="2dp"
        android:bottomRightRadius="2dp"/><!--设置圆角半径-->

    <!--渐变-->
    <gradient
        android:startColor="@android:color/white"
        android:centerColor="@android:color/black"
        android:endColor="@android:color/black"
        android:useLevel="true"
        android:angle="45"
        android:type="radial"
        android:centerX="0"
        android:centerY="0"
        android:gradientRadius="90"/>

    <!--间隔-->
    <padding
        android:left="2dp"
        android:top="2dp"
        android:right="2dp"
        android:bottom="2dp"/><!--各方向的间隔-->

    <!--大小-->
    <size
        android:width="50dp"
        android:height="50dp"/><!--宽度和高度-->

    <!--填充-->
    <solid
        android:color="@android:color/white"/><!--填充的颜色-->

    <!--描边-->
    <stroke
        android:width="2dp"
        android:color="@android:color/black"
        android:dashWidth="1dp"
        android:dashGap="2dp"/>

</shape>
```

(1) 圆角：同时设置五个属性，则 Radius 属性无效。

android:radius="9dp"，设置四个角的半径。

android:topLeftRadius="2dp"，设置左上角的半径。

android:topRightRadius="2dp",设置右上角的半径。

android:bottomLeftRadius="2dp",设置右下角的半径。

android:bottomRightRadius="2dp",设置左下角的半径。

(2) 渐变：当设置填充颜色后,无渐变效果。angle 的值必须是 45 的倍数(包括 0),仅在 type="linear"有效,不然会报错。Angle 对应值的起点如图 4-17 所示。

(3) 间隔：设置四个方向上的间隔。

(4) 大小：设置大小。

(5) 填充：设置填充的颜色。

(6) 描边：dashWidth 和 dashGap 属性,只要其中一个设置为 0,则边框为实线边框。

android:width="2dp",设置边宽。

android:color="@android:color/black",设置边的颜色。

android:dashWidth="1dp",设置虚线的宽度。

android:dashGap="2dp",设置虚线的间隔宽度。

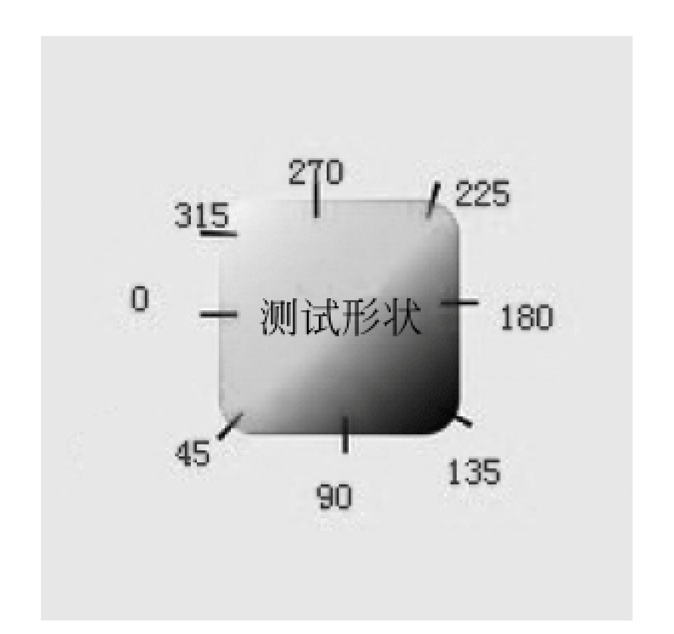


图 4-17 angle 对应值

效果如图 4-18 和图 4-19 所示。



图 4-18 单击按钮前

4.3.2 项目界面相关代码设计

水果连连看的界面设计步骤如下：

(1) 新建“连连看界面”项目。

(2) 在 res 目录下新建 anim 文件夹,放入动画文件。

(3) 在 res 目录下新建 drawable 文件夹,放入图片文件,如图 4-20 所示。



图 4-19　单击按钮后

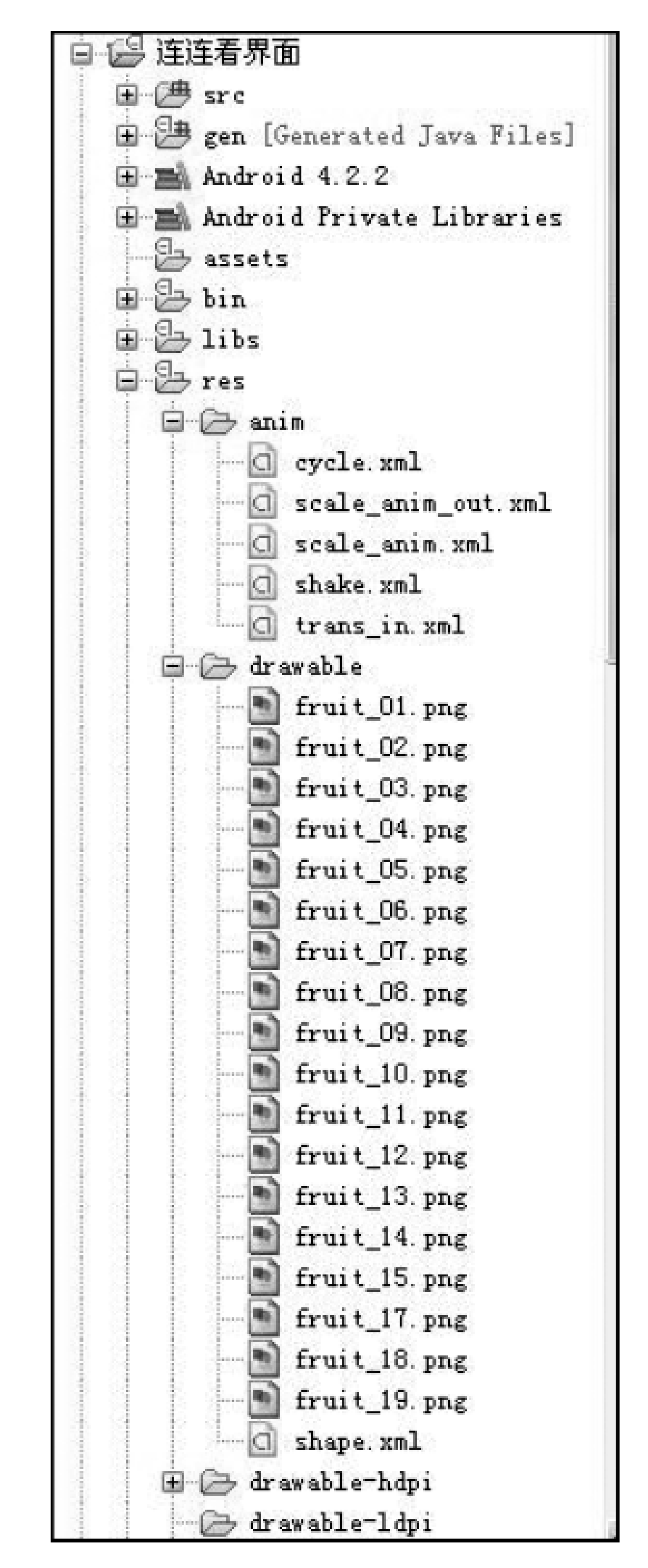


图 4-20　res 目录

(4) 在 src 目录下新建 cn. chap4. link. view 包，在此包中新建 BoardView. java 和 GameView. java，其中 BoardView. java 是游戏棋盘，GameView. java 是游戏控制器。

BoardView. java 主要代码如下：

```
public class BoardView extends View {

    /**
     * xCount x轴方向的图标数+1
     * /
    protected  static final int xCount= 10;
    /**
     * yCount y轴方向的图标数+1
     * /
    protected static final int  yCount= 12;
    /**
     * map 连连看游戏棋盘
     * /
    protected int[][] map= new int[xCount][yCount];
    /**
     * iconSize 图标大小
     * /
    protected int iconSize;
    /**
     * iconCounts 图标的数目
     * /
    protected int iconCounts= 19;
    /**
     * icons 所有的图片
     * /
    protected Bitmap[] icons= new Bitmap[iconCounts];

    /**
     * path 可以连通点的路径
     * /
    private Point[] path= null;
    /**
     * selected 选中的图标
     * /
    protected List< Point> selected= new ArrayList< Point> ();

    public BoardView(Context context,AttributeSet atts) {
        super(context,atts);

        calIconSize();

        Resources r= getResources();
        loadBitmaps(1, r.getDrawable(R.drawable.fruit_01));
        loadBitmaps(2, r.getDrawable(R.drawable.fruit_02));
        loadBitmaps(3, r.getDrawable(R.drawable.fruit_03));
```

```
        loadBitmaps(4, r.getDrawable(R.drawable.fruit_04));
        loadBitmaps(5, r.getDrawable(R.drawable.fruit_05));
        loadBitmaps(6, r.getDrawable(R.drawable.fruit_06));
        loadBitmaps(7, r.getDrawable(R.drawable.fruit_07));
        loadBitmaps(8, r.getDrawable(R.drawable.fruit_08));
        loadBitmaps(9, r.getDrawable(R.drawable.fruit_09));
        loadBitmaps(10, r.getDrawable(R.drawable.fruit_10));
        loadBitmaps(11, r.getDrawable(R.drawable.fruit_11));
        loadBitmaps(12, r.getDrawable(R.drawable.fruit_12));
        loadBitmaps(13, r.getDrawable(R.drawable.fruit_13));
        loadBitmaps(14, r.getDrawable(R.drawable.fruit_14));
        loadBitmaps(15, r.getDrawable(R.drawable.fruit_15));
        loadBitmaps(16, r.getDrawable(R.drawable.fruit_17));
        loadBitmaps(17, r.getDrawable(R.drawable.fruit_18));
        loadBitmaps(18, r.getDrawable(R.drawable.fruit_19));
    }

    /**
     *
     * 计算图标的长宽
     */
    private void calIconSize()
    {
        DisplayMetrics dm=new DisplayMetrics();
        ((Activity) this.getContext()).getWindowManager()
        .getDefaultDisplay().getMetrics(dm);
        iconSize=dm.widthPixels/(xCount);
    }

    /**
     *
     * @param key 特定图标的标识
     * @param d drawable 下的资源
     */
    public void loadBitmaps(int key,Drawable d){
        Bitmap bitmap=Bitmap.createBitmap(iconSize,iconSize,Bitmap.Config.ARGB_8888);
        Canvas canvas=new Canvas(bitmap);
        d.setBounds(0, 0, iconSize, iconSize);
        d.draw(canvas);
        icons[key]=bitmap;
    }

    @Override
    protected void onDraw(Canvas canvas) {

        /**
         * 绘制连通路径,然后将路径以及两个图标清除
         */
        if (path !=null && path.length>=2) {
```

```
                for (int i=0; i<path.length-1; i++) {
                    Paint paint=new Paint();
                    paint.setColor(Color.CYAN);
                    paint.setStyle(Paint.Style.STROKE);
                    paint.setStrokeWidth(3);
                    Point p1=indextoScreen(path[i].x, path[i].y);
                    Point p2=indextoScreen(path[i+1].x, path[i+1].y);
                    canvas.drawLine(p1.x+iconSize / 2, p1.y+iconSize / 2,
                            p2.x+iconSize / 2, p2.y+iconSize / 2, paint);
                }
                Point p=path[0];
                map[p.x][p.y]=0;
                p=path[path.length-1];
                map[p.x][p.y]=0;
                selected.clear();
                path=null;
            }
            /**
             * 绘制棋盘的所有图标,当这个坐标内的值大于0时绘制
             */
            for(int x=0;x<map.length;x+=1){
                for(int y=0;y<map[x].length;y+=1){
                    if(map[x][y]>0){
                        Point p=indextoScreen(x, y);
                        canvas.drawBitmap(icons[map[x][y]], p.x,p.y,null);
                    }
                }
            }

            /**
             * 绘制选中图标,当选中时图标放大显示
             */
            for(Point position:selected){
                Point p=indextoScreen(position.x, position.y);
                if(map[position.x][position.y]>=1){
                    canvas.drawBitmap(icons[map[position.x][position.y]],
                            null,
                            new Rect(p.x-5, p.y-5, p.x+iconSize+5, p.y+iconSize+5), null);
                }
            }
        }

        /**
         *
         * @param path
         */
        public void drawLine(Point[] path) {
            this.path=path;
            this.invalidate();
```

```
    }

    /**
     * 工具方法
     * @param x 数组中的横坐标
     * @param y 数组中的纵坐标
     * @return 将图标在数组中的坐标转成在屏幕上的真实坐标
     */
    public Point indextoScreen(int x,int y){
        return new Point(x* iconSize, y * iconSize);
    }
    /**
     * 工具方法
     * @param x 屏幕中的横坐标
     * @param y 屏幕中的纵坐标
     * @return 将图标在屏幕中的坐标转成在数组上的虚拟坐标
     */
    public Point screenToindex(int x,int y){
        int ix=x/iconSize;
        int iy=y/iconSize;
        if(ix<xCount && iy<yCount){
            return new Point( ix,iy);
        }else{
            return new Point(0,0);
        }
    }
}
```

GameView.java 主要代码如下：

```
//游戏控制器继承游戏视图
public class GameView extends BoardView
//初始化游戏视图
public void initMap() {
        int x=1;
        int y=0;
        for (int i=1; i<xCount-1; i++) {
            for (int j=1; j<yCount-1; j++) {
                map[i][j]=x;
                if (y==1) {
                    x++;
                    y=0;
                    if (x==iconCounts) {
                        x=1;
                    }
                } else {
                    y=1;
                }
        }
    }
```

```
}
```

（5）在 layout 目录下新建 welcome. xml，游戏的启动页面，如图 4-21 所示。



图 4-21 游戏启动页面

当单击开始按钮时此页面替换成游戏界面。

主要代码如下：

```
<cn.chap4.link.view.GameView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/game_view"
    android:visibility="gone"
    android:layout_below="@id/timer"
/>
```

（6）在 src 目录下的 cn. chap4. link 包中新建 WelActivity. java，把如图 4-22 所示的页面设置为启动页面，添加单击事件，Java 代码如下：

```
setContentView(R.layout.welcome);
btnPlay= (ImageButton) findViewById(R.id.play_btn);
btnPlay.setOnClickListener(this);
@Override
    public void onClick(View v) {

        switch(v.getId()){
        case R.id.play_btn:
            Animation scaleOut=
            AnimationUtils.loadAnimation(this,R.anim.scale_anim_out);
            Animation transIn=
            AnimationUtils.loadAnimation(this,R.anim.trans_in);
```

```
                btnPlay.startAnimation(scaleOut);
                btnPlay.setVisibility(View.GONE);
                imgTitle.setVisibility(View.GONE);
                gameView.setVisibility(View.VISIBLE);

                btnRefresh.setVisibility(View.VISIBLE);
                btnTip.setVisibility(View.VISIBLE);
                progress.setVisibility(View.VISIBLE);
                clock.setVisibility(View.VISIBLE);
                textRefreshNum.setVisibility(View.VISIBLE);
                textTipNum.setVisibility(View.VISIBLE);

                btnRefresh.startAnimation(transIn);
                btnTip.startAnimation(transIn);
                gameView.startAnimation(transIn);
                gameView.startPlay();
                break;
            }
        }
```

（7）在 res 目录下新建 dialog_view. xml，在这个布局文件中设置下一关、重玩本关等功能，如图 4-22 所示。

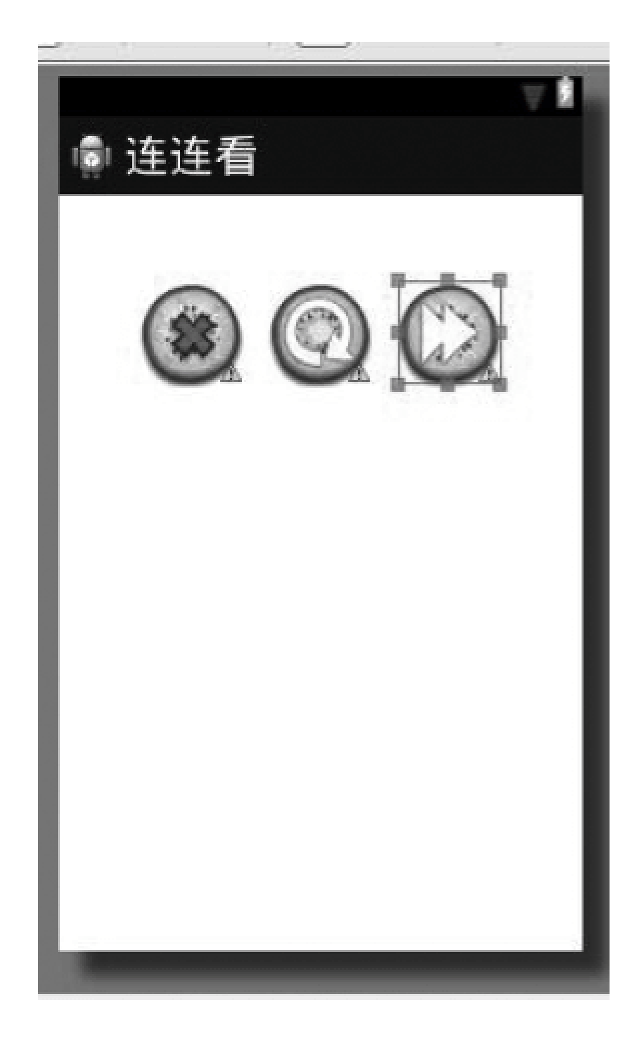


图 4-22　在布局文件中设置功能

4.4　项目功能的实现

代码编写实现的是应用程序中的逻辑代码功能，该部分代码在项目的 src 目录中完成，如图 4-23 所示。

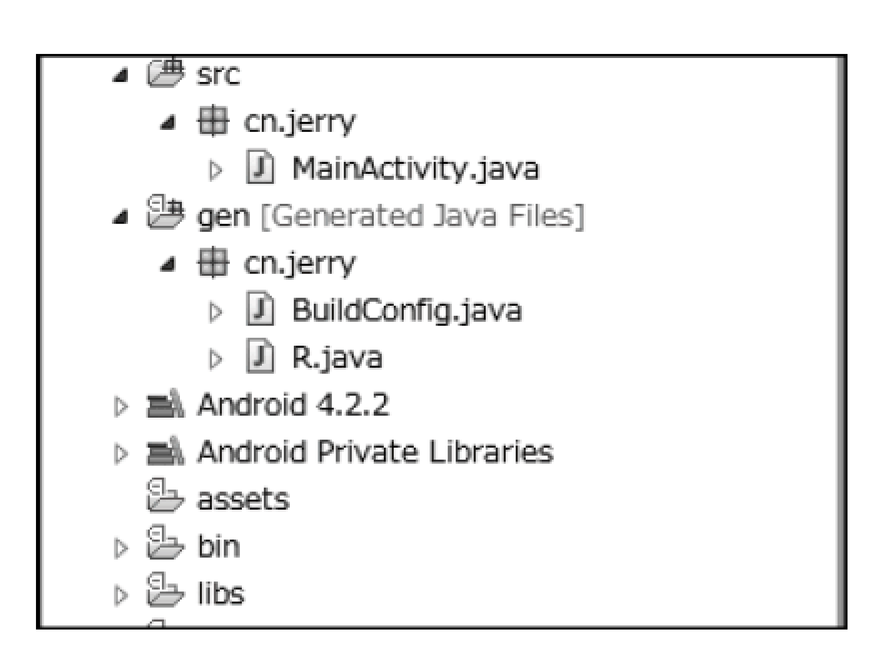


图 4-23 代码编写

4.4.1 知识准备

在 Android 中，有很多功能是不能放在 onCreate 或者 onStart 方法里面的，因为这些功能耗时相对较长。比如一个文件下载的过程比较长，但是如果写在 Activity 中，这段时间内 Activity 是完全没有响应的，那么就可以将这种处理大量数据或者耗时比较长的功能放在一个单独的线程中完成，即 Activity 是一个线程，而下载的是在另外一个线程，这样就可以使下载与 Activity 之间互不影响，从而得到良好的用户体验。这里有两种队列：一种是线程队列，就是用 postXX 方法或者 removeCallbacks 方法对线程对象的操作；另一种是消息队列，用 sendMessage 和 handleMessage 方法对消息对象进行处理。队列如图 4-24 所示。



图 4-24 队列

handler 采用的是一个消息队列的方式，每一个 handler 都有一个与之关联的消息队列，而且是按照先进先出的方式执行，即每次加入一个 handler，然后拿出来对其进行处理，之后再拿出另一个，再进行处理。

例 4-1 仅仅对线程对象进行操作的测试。

Java 代码如下：

```
private Button mybutton1;
private Button mybutton2;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    mybutton1= (Button)findViewById(R.id.button1);
```

```
        mybutton2= (Button)findViewById(R.id.button2);

        mybutton1.setOnClickListener(new Button.OnClickListener()
        {

            @Override
            public void onClick(View arg0) {
                /**
                 * 调用 Handler 的 post 方法,将要执行的线程对象添加到
                 * 线程队列中
                 * /
                handler.post(updateThread);
            }

        });
        mybutton2.setOnClickListener(new Button.OnClickListener()
        {

            @Override
            public void onClick(View v) {
                //TODO Auto-generated method stub
                handler.removeCallbacks(updateThread);
            }

        });

    }
    //创建 Handler 对象
    Handler handler=new Handler();
    /**
    * 将要执行的操作卸载写入线程对象的 run()方法中
    * /
    Runnable updateThread=new Runnable()
    {
        public void run()
        {
            System.out.println("更新线程");
            //在 run 方法内部,执行 postXX 的方法,每隔 3 秒会执行一次
            handler.postDelayed(updateThread, 3000);
        }
    };
```

运行结果如图 4-25 所示。

程序解释：首先创建一个 Handler 对象，然后创建一个继承自 Runnable 接口的线程。

程序首先单击“开始”按钮，会马上执行 post 方法，将执行的线程对象添加到线程队

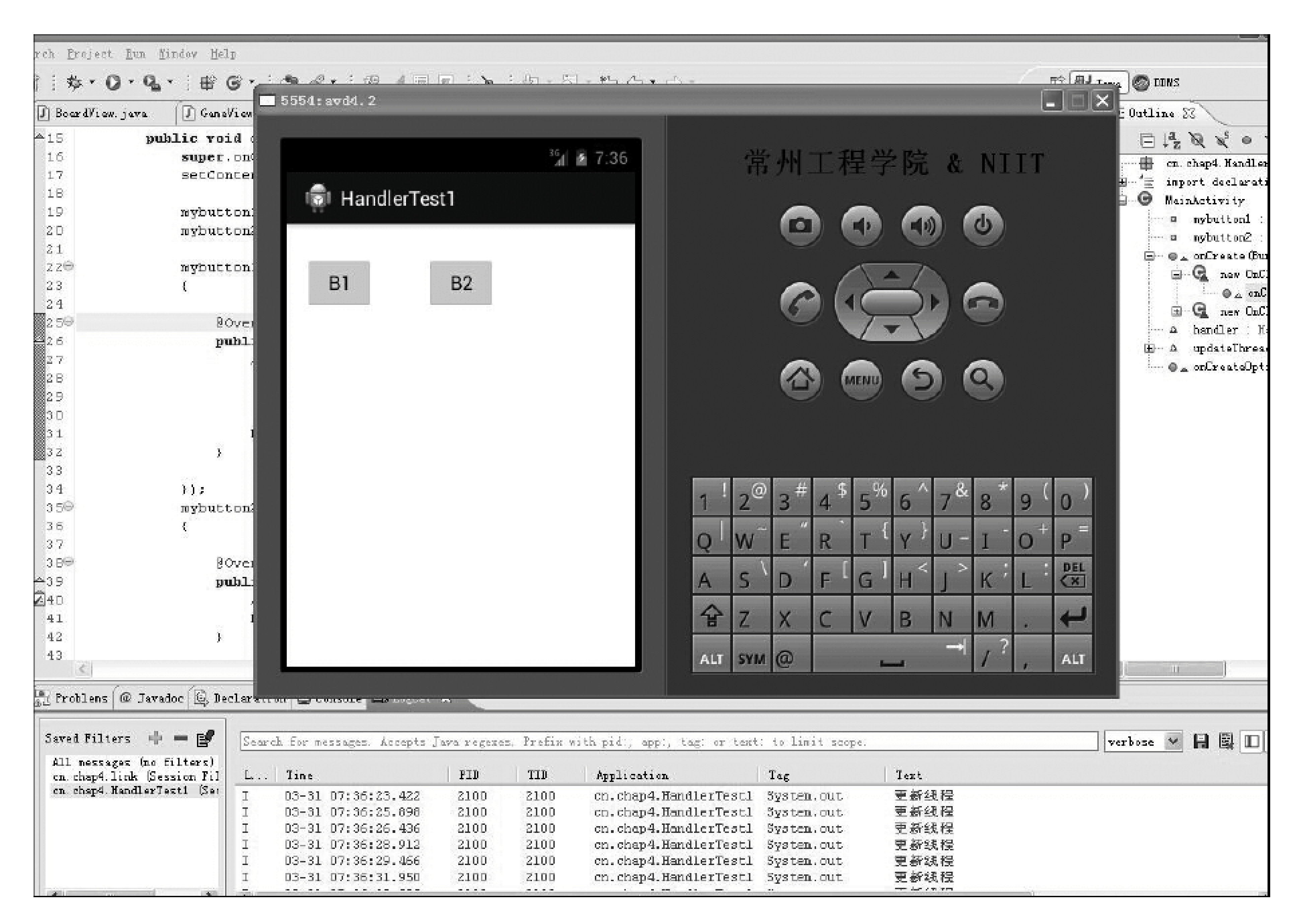


图 4-25 Handler 对象操作

列中,这时会马上开始执行。

Java 代码如下:

```
public void run()
    {
        System.out.println("更新线程");
        //在 run 方法内部,执行 postXX 的方法,每隔 3 秒会执行一次
        handler.postDelayed(updateThread, 3000);
    }
```

然后,执行 postDelayed 方法,依据其中设置的间隔时间,每 3 秒会调用一个 Handler 对象到线程队列中,并且一直执行,直到单击“结束”按钮,调用 removeCallbacks 方法将其从线程队列中移除。

例 4-2 简单地对线程对象和消息对象进行处理。

Java 代码如下:

```
private ProgressBar bar=null;
    private Button start=null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        bar=(ProgressBar)findViewById(R.id.progress1);
        start=(Button)findViewById(R.id.start);
        start.setOnClickListener(new Button.OnClickListener()
```

```
        {
            @Override
            public void onClick(View v) {
                bar.setVisibility(View.VISIBLE);
                handler.post(handlerThread);
            }

        });
}
/**
  * 使用匿名内部类复写 hanlder 中的 hanldrMessage 方法
  * 这里的 msg 对象就是从线程部分发送过来的对象
  * /
Handler handler=new Handler()
{
    public void handleMessage(Message msg)
    {
        bar.setProgress(msg.arg1);
        handler.post(handlerThread);
    }
};
//线程类,该类使用的是匿名内部类的方式进行声明
Runnable handlerThread=new Runnable()
{
    int i=0;
    public void run()
    {
        System.out.println("开始线程");
        i=i+10;
        /**
          * 得到一个消息对象,Message 类是由 Android 操作系统提供
          * obtainMessage 方法用来得到 Message 对象
          * /
        Message msg=handler.obtainMessage();
        /**
          * Message 中有成员变量,即 msg 独享的 arg1 参数
          * 将其值设置为 i。用 arg1 或 arg2 这两个成员变量传递
          * 消息,优点是系统性能消耗较少
          * /
        msg.arg1=i;
        try {
            //当前线程休眠 1 秒
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            //TODO Auto-generated catch block
            e.printStackTrace();
        }
        /**
          * 发送一个消息,用 sendMessage 是将 msg 加入消息
```

```
            * 队列中,而 post 是将线程加入线程队列
            */
        handler.sendMessage(msg);
        if(i==100)
        {
            /**
              * 如果 i=100时,就将线程对象
              * 从 handler 中移除
              */
            handler.removeCallbacks(handlerThread);
            bar.setVisibility(View.GONE);
        }
    }
};
```

activity_main.xml 代码如下:

```
<ProgressBar
    android:id="@+id/progress1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:visibility="gone"
    style="?android:attr/progressBarStyleHorizontal"
/>

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="27dp"
        android:layout_marginTop="90dp"
        android:text="点我" />
```

运行结果如图 4-26 和图 4-27 所示。

程序说明:

(1) 单击按钮后,会执行按钮的 onClick 方法中的 LickListener,将进度条显示出来,并且将线程对象加入线程队列。

Java 代码如下:

```
bar.setVisibility(View.VISIBLE);
handler.post(handlerThread);
```

(2) 线程对象先打印出一个“开始线程”,然后 i 的值增加 10,再从系统中获取一个 Message 对象。

(3) 将 i 赋给 Message 对象的参数 arg1。

(4) 当前线程休眠 5 秒,然后通过 sendMessage 方法发送一个 Message 对象到消息队列中。



图 4-26　线程对象及消息对象处理 1



图 4-27　线程对象及消息对象处理 2

(5) 再执行,通过 handleMessage 方法设置进度条的值,并且将其加入进程队列。

Java 代码如下:

```
Handler handler= new Handler()
    {
        public void handleMessage(Message msg)
```

```
        {
            bar.setProgress(msg.arg1);
            handler.post(handlerThread);
        }
    };
```

(6) 循环执行,直到 i=100,隐藏进度条,并将线程对象从线程队列中移除。

4.4.2 项目功能相关代码设计

1. 设置监听器

(1) 编写状态改变监听器 OnStateListener。

Java 代码如下:

```
public interface OnStateListener{
    public void OnStateChanged(int StateMode);
}
```

(2) 编写倒计时监听器 OnTimerListener。

Java 代码如下:

```
public interface OnTimerListener{
    public void onTimer(int leftTime);
}
```

(3) 编写工具状态改变监听器 OnToolsChangeListener。

Java 代码如下:

```
public interface OnToolsChangeListener{
    public void onRefreshChanged(int count);
    public void onTipChanged(int count);
}
```

2. 编写控制器功能

```
    //设置 5种状态
    public static final int WIN=1;                  //赢
    public static final int LOSE=2;                 //失败
    public static final int PAUSE=3;                //暂停
    public static final int PLAY=4;                 //开始游戏
    public static final int QUIT=5;                 //退出

//开始游戏方法
public void startPlay(){
        Help=3;
        Refresh=3;
        isStop=false;
        toolsChangedListener.onRefreshChanged(Refresh);
        toolsChangedListener.onTipChanged(Help);
```

```
            leftTime= totalTime;
            initMap();
            refreshTime= new RefreshTime();
            refreshTime.start();
            GameView.this.invalidate();
    }
    public void startNextPlay(){
            //下一关为上一关减去 10秒的时间
            totalTime-=10;
            startPlay();
    }
    //判断是否赢
    private boolean win() {
            for (int x= 0; x< xCount; x++) {
                for (int y= 0; y< yCount; y++) {
                    if (map[x][y] != 0) {
                        return false;
                    }
                }
            }
            return true;
    }
    //判断是否无解
    private boolean die() {
            for (int y= 1; y< yCount- 1; y++) {
                for (int x= 1; x< xCount- 1; x++) {
                    if (map[x][y] != 0) {
                        for (int j= y; j< yCount- 1; j++) {
                            if (j== y) {
                                for (int i= x+ 1; i< xCount- 1; i++) {
                                    if (map[i][j]== map[x][y]
                                            && link(new Point(x, y),
                                                    new Point(i, j))) {
                                        return false;
                                    }
                                }
                            } else {
                                for (int i= 1; i< xCount- 1; i++) {
                                    if (map[i][j]== map[x][y]
                                            && link(new Point(x, y),
                                                    new Point(i, j))) {
                                        return false;
                                    }
                                }
                            }
                        }
                    }
                }
```

```
        }
        return true;
}
//调换水果位置
private void change() {
        Random random=new Random();
        int tmpV, tmpX, tmpY;
        for (int x=1; x<xCount-1; x++) {
            for (int y=1; y<yCount-1; y++) {
                tmpX=1+random.nextInt(xCount-2);
                tmpY=1+random.nextInt(yCount-2);
                tmpV=map[x][y];
                map[x][y]=map[tmpX][tmpY];
                map[tmpX][tmpY]=tmpV;
            }
        }
        if (die()) {
            change();
        }
        GameView.this.invalidate();
}

//使用线程实现倒计时
class RefreshTime extends Thread {

        public void run() {
            while (leftTime>=0 && !isStop) {
                timerListener.onTimer(leftTime);
                leftTime--;
                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
            if(!isStop){
                setMode(LOSE);
            }

        }
}
//handler实现刷新界面
class RefreshHandler extends Handler {

        @Override
        public void handleMessage(Message msg) {
            super.handleMessage(msg);
            if (msg.what==REFRESH_VIEW) {
                GameView.this.invalidate();
```

```
                if (win()) {
                    setMode(WIN);
                    isStop= true;
                } else if (die()) {
                    change();
                }
            }
        }

        public void sleep(int delayTime) {
            this.removeMessages(0);
            Message message= new Message();
            message.what= REFRESH_VIEW;
            sendMessageDelayed(message, delayTime);
        }
}
```

//连连看算法实现,这里分 3种情况,同前面算法中一样。

```
List< Point> p1E= new ArrayList< Point> ();
    List< Point> p2E= new ArrayList< Point> ();

    private boolean link(Point p1, Point p2) {
        if (p1.equals(p2)) {
            return false;
        }
        path.clear();
        if (map[p1.x][p1.y]==map[p2.x][p2.y]) {
            if (linkD(p1, p2)) {
                path.add(p1);
                path.add(p2);
                return true;
            }

            Point p= new Point(p1.x, p2.y);
            if (map[p.x][p.y]==0) {
                if (linkD(p1, p) && linkD(p, p2)) {
                    path.add(p1);
                    path.add(p);
                    path.add(p2);
                    return true;
                }
            }
            p= new Point(p2.x, p1.y);
            if (map[p.x][p.y]==0) {
                if (linkD(p1, p) && linkD(p, p2)) {
                    path.add(p1);
                    path.add(p);
                    path.add(p2);
                    return true;
```

```
                }
            }
            expandX(p1, p1E);
            expandX(p2, p2E);

            for (Point pt1 : p1E) {
                for (Point pt2 : p2E) {
                    if (pt1.x==pt2.x) {
                        if (linkD(pt1, pt2)) {
                            path.add(p1);
                            path.add(pt1);
                            path.add(pt2);
                            path.add(p2);
                            return true;
                        }
                    }
                }
            }

            expandY(p1, p1E);
            expandY(p2, p2E);
            for (Point pt1 : p1E) {
                for (Point pt2 : p2E) {
                    if (pt1.y==pt2.y) {
                        if (linkD(pt1, pt2)) {
                            path.add(p1);
                            path.add(pt1);
                            path.add(pt2);
                            path.add(p2);
                            return true;
                        }
                    }
                }
            }
            return false;
        }
        return false;
    }

//判断两个棋子是否能直接相连,分为横连和竖连。
private boolean linkD(Point p1, Point p2) {
        if (p1.x==p2.x) {
            int y1=Math.min(p1.y, p2.y);
            int y2=Math.max(p1.y, p2.y);
            boolean flag=true;
            for (int y=y1+1; y<y2; y++) {
                if (map[p1.x][y] !=0) {
                    flag=false;
                    break;
```

```
                }
            }
            if (flag) {
                return true;
            }
        }
        if (p1.y==p2.y) {
            int x1=Math.min(p1.x, p2.x);
            int x2=Math.max(p1.x, p2.x);
            boolean flag=true;
            for (int x=x1+1; x<x2; x++) {
                if (map[x][p1.y] != 0) {
                    flag=false;
                    break;
                }
            }
            if (flag) {
                return true;
            }
        }
        return false;
}
//水平扫描延伸和垂直扫描延伸
private void expandX(Point p, List<Point> l) {
    l.clear();
    for (int x=p.x+1; x<xCount; x++) {
        if (map[x][p.y] != 0) {
            break;
        }
        l.add(new Point(x, p.y));
    }
    for (int x=p.x-1; x>=0; x--) {
        if (map[x][p.y] != 0) {
            break;
        }
        l.add(new Point(x, p.y));
    }
}

private void expandY(Point p, List<Point> l) {
    l.clear();
    for (int y=p.y+1; y<yCount; y++) {
        if (map[p.x][y] != 0) {
            break;
        }
        l.add(new Point(p.x, y));
    }
    for (int y=p.y-1; y>=0; y--) {
        if (map[p.x][y] != 0) {
```

```
                break;
            }
            l.add(new Point(p.x, y));
        }
}
```

3. 在 WelActivity.java 中实现以下功能

```
//开始游戏、提示、重新排列按钮单击事件
@Override
public void onClick(View v) {

    switch(v.getId()){
    case R.id.play_btn:
        Animation scaleOut=
        AnimationUtils.loadAnimation(this,R.anim.scale_anim_out);
        Animation transIn=
        AnimationUtils.loadAnimation(this,R.anim.trans_in);

        btnPlay.startAnimation(scaleOut);
        btnPlay.setVisibility(View.GONE);
        imgTitle.setVisibility(View.GONE);
        gameView.setVisibility(View.VISIBLE);

        btnRefresh.setVisibility(View.VISIBLE);
        btnTip.setVisibility(View.VISIBLE);
        progress.setVisibility(View.VISIBLE);
        clock.setVisibility(View.VISIBLE);
        textRefreshNum.setVisibility(View.VISIBLE);
        textTipNum.setVisibility(View.VISIBLE);

        btnRefresh.startAnimation(transIn);
        btnTip.startAnimation(transIn);
        gameView.startAnimation(transIn);
        gameView.startPlay();
        break;
    case R.id.refresh_btn:
        Animation shake01=AnimationUtils.loadAnimation(this,R.anim.shake);
        btnRefresh.startAnimation(shake01);
        gameView.refreshChange();
        break;
    case R.id.tip_btn:
        Animation shake02=AnimationUtils.loadAnimation(this,R.anim.shake);
        btnTip.startAnimation(shake02);
        gameView.autoClear();
        break;
    }
}

//倒计时更新进度条
```

```
@Override
public void onTimer(int leftTime) {
    Log.i("onTimer", leftTime+ "");
    progress.setProgress(leftTime);
}
//游戏状态改变
@Override
public void OnStateChanged(int StateMode) {
    switch(StateMode){
    case GameView.WIN:
        handler.sendEmptyMessage(0);
        break;
    case GameView.LOSE:
        handler.sendEmptyMessage(1);
        break;
    case GameView.PAUSE:
        gameView.stopTimer();
        break;
    case GameView.QUIT:
        gameView.stopTimer();
        break;
    }
}
//刷新剩余次数、提示剩余次数
@Override
    public void onRefreshChanged(int count) {
        textRefreshNum.setText(""+ gameView.getRefreshNum());
    }

    @Override
    public void onTipChanged(int count) {
        textTipNum.setText(""+ gameView.getTipNum());
    }
//退出
public void quit(){
        this.finish();
}
```

4. 设置按钮事件

在 cn.chap4.link 包下增加自定义对话框 MyDialog.java，引用 dialog_view.java 布局文件。设置菜单、重玩本关和下一关按钮事件。代码如下：

```
@Override
    public void onClick(View v) {
        this.dismiss();
        switch(v.getId()){
        case R.id.menu_imgbtn:
            Dialog dialog= new AlertDialog.Builder(context)
```

```
                .setIcon(R.drawable.buttons_bg20)
                .setTitle(R.string.quit)
                .setMessage(R.string.sure_quit)
                .setPositiveButton(R.string.alert_dialog_ok,
                new DialogInterface.OnClickListener() {
                  public void onClick(DialogInterface dialog, int whichButton) {
                      ((WelActivity)context).quit();
                  }
                })
                .setNegativeButton(R.string.alert_dialog_cancel,
                new DialogInterface.OnClickListener() {
                  public void onClick(DialogInterface dialog, int whichButton) {
                  }
              })
                .create();
                dialog.show();
                break;
        case R.id.replay_imgbtn:
              gameview.startPlay();
              break;
        case R.id.next_imgbtn:
              gameview.startNextPlay();
              break;
        }
    }
```

4.5 系统运行与效果测试

系统运行与效果测试如图 4-28～图 4-32 所示。



图 4-28 系统运行与效果测试 1



图 4-29　系统运行与效果测试 2



图 4-30　系统运行与效果测试 3



图 4-31　系统运行与效果测试 4



图 4-32　系统运行与效果测试 5

4.6　本章小结

通过本章项目练习，学习了 Android 的图片控件、滑块控件、自定义 View、动画和渐变等效果，连连看是一款经典的益智游戏，感兴趣的朋友可以在此基础上继续扩展，创建更好玩的游戏。

4.7 项目实践

（1）把做好的 APP 安装到实验用手机进行测试。

（2）练习自定义控件的实现。

（3）灵活使用 Shape 自定义控件背景。

（4）扩展动画效果。

（5）理解 Android 中的 Handler。

第 5 章

聊天工具的设计及开发

本章的工作目标如下：

(1) 实现 QQ 登录功能。

(2) 实现 QQ 好友列表功能。

(3) 实现 QQ 聊天功能。

5.1 项 目 分 析

本项目的学习和操作主要关注以下几点。

(1) 掌握 RadioGroup 和 RadioButton、CheckBox、ProgressBar、ListView、ExpandableListActivity 等常用控件的使用。

(2) 在 QQ 登录界面中，单击下拉框选择 QQ 账号，实现登录功能。

(3) 在 QQ 好友界面中，使用 ListView 和 ExpandableListActivity 混合布局，添加单击事件。

(4) 在聊天界面中，模拟 QQ 聊天。

QQ 登录界面由账号、密码、登录按钮等常用控件组成，账号下拉选择使用 ListView 控件的隐藏于显示实现，单击登录按钮，跳转到 QQ 好友列表界面，好友列表由 ListView 和 ExpandableListActivity（二级列表）控件组成，ExpandableListActivity 控件显示 QQ 好友分组及每组中的好友信息，单击分组展开组中的好友，单击好友跳转到 QQ 聊天界面，在聊天界面中模拟 2 个人聊天功能，把聊天信息显示在界面上。

5.2 项目界面设计

5.2.1 知识准备

1. RadioGroup 和 RadioButton 的使用方法

RadioGroup 和 RadioButton 的关系如下：

(1) RadioButton 表示单个圆形单选框，而 RadioGroup 是可以容纳多个 RadioButton 的容器。

（2）每个 RadioGroup 中的 RadioButton 同时只能有一个被选中。

（3）不同的 RadioGroup 中的 RadioButton 互不相干，即如果组 A 中有一个被选中了，组 B 中依然可以有一个被选中。

（4）大部分场合下，一个 RadioGroup 中至少有 2 个 RadioButton。

（5）大部分场合下，一个 RadioGroup 中的 RadioButton 默认会有一个被选中，并建议用户将它放在 RadioGroup 中的起始位置。

XML 布局代码如下：

```
<RadioGroup
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:orientation="horizontal"
        >

        <RadioButton
            android:id="@+id/radioButton1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="男"
            android:checked="true"
            />

        <RadioButton
            android:id="@+id/radioButton2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="女" />

</RadioGroup>
```

其中 android:checked="true"表示该选项被选中，如图 5-1 所示。

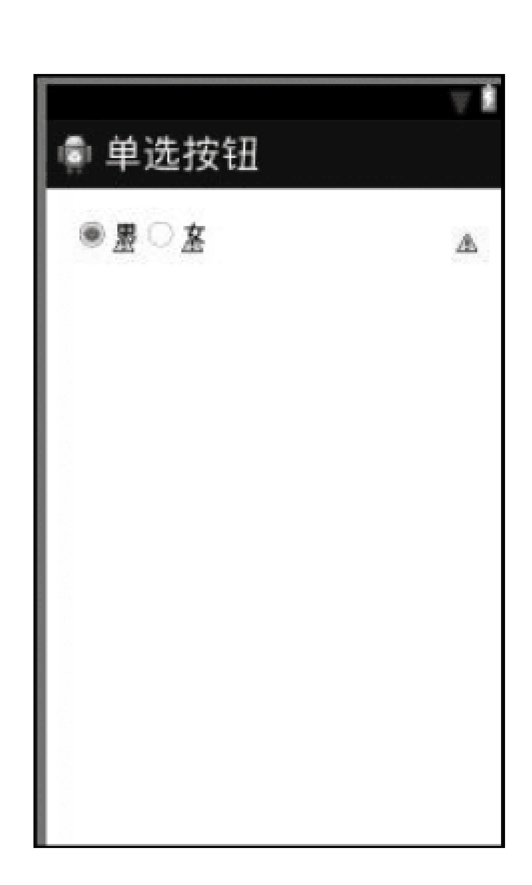


图 5-1　选中选项

选中项变更的事件监听：OnCheckedChangeListener。

代码如下：

```
//处理单选按钮是否选中事件
    OnCheckedChangeListener listener=new OnCheckedChangeListener() {
        /**
         * 第一个参数:当前选中的按钮
         * 第二个参数:是否被选中
         *
         * /
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean
        isChecked) {
            //把组件按钮强制转换成当前单击的单选按钮
            RadioButton rb=(RadioButton) buttonView;
```

```
            if(rb.isChecked()){
                Toast.makeText(MainActivity.this, rb.getText()+"被选中", 1).show();
            }
        }
};
```

效果如图 5-2 所示。



图 5-2　单选按钮效果图

2. CheckBox 的使用方法

CheckBox 和 Button 一样，也是一种古老的控件。它的优点在于，不需用户填写具体信息，只需轻轻单击。缺点在于只有“是”和“否”两种情况。我们往往就是利用它的这个特性获取用户的一些信息。

XML 布局代码如下：

```
<CheckBox
            android:id="@+id/checkBox1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:layout_marginLeft="19dp"
            android:layout_marginTop="20dp"
            android:text="编程"
            android:checked="true" />

    <CheckBox
            android:id="@+id/checkBox2"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/checkBox1"
        android:layout_below="@+id/checkBox1"
        android:layout_marginTop="22dp"
        android:text="打球" />

    <CheckBox
        android:id="@+id/checkBox3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/checkBox2"
        android:layout_below="@+id/checkBox2"
        android:layout_marginTop="27dp"
        android:text="唱歌" />
```

布局如图 5-3 所示。

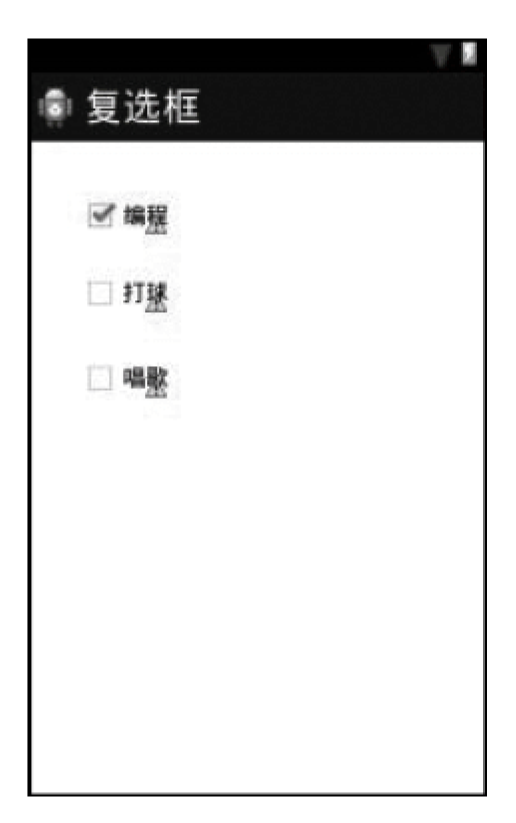


图 5-3　复选框布局图

用户触摸它的改变将会触发 OnCheckedChange 事件，而用户可以对应地使用 OnCheckedChangeListener 监听器监听这个事件。

单击事件代码如下：

```
OnCheckedChangeListener listener=new OnCheckedChangeListener() {

        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean
        isChecked) {
            //强制转换成复选框
            CheckBox cb=(CheckBox) buttonView;
            if(isChecked)
                Toast.makeText(MainActivity.this, cb.getText()+"被选中...", 0).show();
            else
                Toast.makeText(MainActivity.this, cb.getText()+"未选中...", 0).show();
        }
};
```

效果如图 5-4 所示。



图 5-4　复选框效果图

3．ProgressBar 的使用方法

在某些操作的进度中的可视指示器，为用户呈现操作的进度，它还有一个次要的进度条，用来显示中间进度，如在流媒体播放的缓冲区的进度。一个进度条也可不确定其进度。在不确定模式下，进度条显示循环动画。这种模式常用于应用程序使用未知的任务长度的情况。

（1）重要属性

① android:progressBarStyle：默认进度条样式。

② android:progressBarStyleHorizontal：水平样式。

（2）重要方法

① getMax()：返回这个进度条的范围的上限。

② getProgress()：返回进度。

③ getSecondaryProgress()：返回次要进度。

④ incrementProgressBy(int diff)：指定增加的进度。

⑤ isIndeterminate()：指示进度条是否在不确定模式下。

⑥ setIndeterminate(boolean indeterminate)：设置在不确定模式下。

⑦ setVisibility(int v)：设置该进度条是否可视。

（3）重要事件

onSizeChanged(int w，int h，int oldw，int oldh)：当进度值改变时引发此事件。

XML 代码如下：

```
<ProgressBar
    android:id="@+id/progressBar1"
    style="?android:attr/progressBarStyleLarge"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_marginLeft="24dp"
    android:layout_marginTop="20dp" />

<ProgressBar
    android:id="@+id/progressBar2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/progressBar1"
    android:layout_below="@+id/progressBar1"
    android:layout_marginTop="24dp" />
<!--android:max="100" 进度值,默认 100-->
<ProgressBar
    android:id="@+id/progressBar3"
    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/progressBar2"
    android:layout_centerVertical="true"
    android:max="100"/>
```

进度条布局如图 5-5 所示。

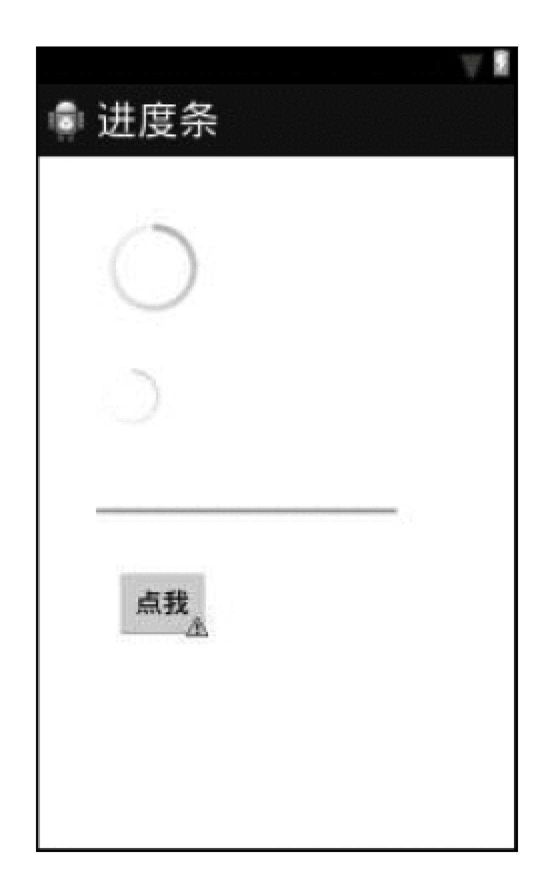


图 5-5　进度条效果图 1

代码如下：

```
int count=0;
public void clickMe(View view){
        count+=10;
```

```
        //1.获取进度条
        ProgressBar pb= (ProgressBar) findViewById(R.id.progressBar3);
        //2.设置进度值
        pb.setProgress(count);
        //3.设置第二进度条
        pb.setSecondaryProgress(count+ 8);
}
```

效果如图 5-6 所示。



图 5-6　进度条效果图 2

当用户单击“点我”按钮时，是想让进度条持续不断地更新，这是一长耗时的操作，处理不好会导致系统假死，用户体验很差。而 Android 则更进一步，如果任意一个 Acitivity 5 秒以上没有响应，就会被强制关闭，因此需要另外启动一个线程处理长耗时的操作，而主线程则不受其影响，在耗时操作完结后发送消息给主线程，主线程再做相应处理。线程之间的消息传递和异步处理用的就是 Handler。

Thread 线程发出 Handler 消息，通知更新 UI。Handler 根据接收的消息，处理 UI 更新。此时，我们可以使用 Handler 实现单击“点我”按钮，让进度条持续不断地更新。

代码如下：

```
public void go(View view){
        //启动线程
        new Thread(runnable).start();

}
    //消息机制,更新主界面
    Handler handler= new Handler(){
        //接收消息,更新界面
```

```
        @Override
        public void handleMessage(Message msg) {
            //2.设置进度
            pb.setProgress(count);
        }
    };

    //创建一个线程处理业务逻辑
    Runnable runnable=new Runnable() {

        @Override
        public void run() {
            while(count<100){
                count+=10;
                //发送消息
                handler.sendEmptyMessage(0);
                try {
                    Thread.sleep(500);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    };
```

4. Spinner 的基本用法

Spinner 是一个列表选择框，在用户选择后，会展示一个列表供用户进行选择。Spinner 是 ViewGroup 的间接子类，它和其他的 Android 控件一样，数据需要使用 Adapter 进行封装。

Spinner 的常用 XML 属性如下。

- android:spinnerMode：列表显示的模式，有两个选择，为弹出列表(dialog)以及下拉列表(dropdown)，如果不特别设置，为下拉列表。
- android:entries：使用资源配置数据源。
- android:prompt：对当前下拉列表设置标题，仅在 dialog 模式下有效。传递一个 @string/name 资源，需要在资源文件中定义。

Spinner 支持的常用事件有以下几个。

- AdapterView.OnItemClickListener：列表项被单击时触发。
- AdapterView.OnItemLongClickListener：列表项被长按时触发。
- AdapterView.OnItemSelectedListener：列表项被选择时触发。

因为适配器可以设置各种不同的样式，有选择、单选、多选，所以 OnItemClickListener 和 OnItemSelectedListener 适用于不同的场景。

(1) Spinner 的数据绑定

对于 Spinner 展示的数据源，一般使用两种方式设定数据：①通过 XML 资源文件设

置，这种方式比较死板，也比较直观；②使用 Adapter 接口设置，这是最常见的方式，动态、灵活，可以设定各种样式以及数据来源。

通过 XML 资源文件设置 Spinner 数据的方式，首先需要在/res/values 目录下新建 XML 格式的资源文件，一般会使用 strings. xml。在其中定义标签，通过标签设置选择数据。

XML 代码如下：

```
<string-array name="users">
    <item>张飞</item>
    <item>关羽</item>
    <item>赵云</item>
    <item>曹操</item>
    <item>司马懿</item>
</string-array>
```

通过适配器 Adapter 可以设定比较复杂的展示效果，一般项目中比较常用的也是这种方式。但是如果对于动态的、简单的数据，可以使用 ArrayAdapter 对象设置适配器。

Java 代码如下：

```
//创建数据适配器,设置数据和样式
        //第一个参数:上下文应用
        //第二个参数:文本数组的 id
        //第三个参数:下拉框的样式
        ArrayAdapter adapter=ArrayAdapter.createFromResource(this,
                R.array.users, android.R.layout.simple_spinner_item);
        //设置下拉时每个选项的样式

        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        //绑定数据
        Spinner spinner=(Spinner) findViewById(R.id.spinner1);
        spinner.setAdapter(adapter);
        spinner.setPrompt("三国人物");
        //绑定监听器
        spinner.setOnItemSelectedListener(listener);

//选中某个选项的监听器
    OnItemSelectedListener listener=new OnItemSelectedListener() {
        //第一个参数:spinner 控件
        //第二个参数:每个选项控件
        //第三个参数:选中选项的位置
        //第四个参数:组件 id
        @Override
        public void onItemSelected(AdapterView<?> parent, View view,
                int position, long id) {
            String user=parent.getItemAtPosition(position).toString();
            Toast.makeText(MainActivity.this, user, 0).show();
        }

        @Override
```

```
        public void onNothingSelected(AdapterView<?> parent) {

        }
    };
```

效果如图 5-7 所示。



图 5-7 下拉框效果图

(2) SimpleAdapter 配置 Spinner 数据

对于一个稍复杂的数据的展示，ArrayAdapter 是无法满足需求的，下面介绍 SimpleAdapter，它同样继承自 Adapter。

SimpleAdapter 是一个简单的适配器，映射静态的 XML 格式的布局文件到视图中。可以指定一个 List<Map<P,T>>格式的数据，List 中的每一条数据对应一行，而 Map 中的每一条数据对应数据行的一列。这个数据用来映射到 XML 定义的布局控件中，对应关系通过构造函数的另外两个参数指定，下面介绍一下 SimpleAdapter 的构造函数。

```
SimpleAdapter(Context context, List<? extends Map<String, ?>> data, int resource, String[] from, int[]
to)
```

构造函数的含义如下。

- context：上下文对象，一般就是当前的 Activity。
- data：上面介绍的 List<Map<S,T>>类型的数据。
- resource：XML 资源的 ID，通过 R 对象选中。
- from：一个 String 类型数组，每条数据对应 data 数据中，Map 结构定义的 Key。
- to：一个 int 类型数组，对应 XML 资源中控件的 ID，注意顺序必须与 from 中指定数据的顺序一致。

下面通过一个示例讲解一下 SimpleAdapter 是如何设置自定义格式数据的。

布局代码如下：

```
<Spinner
        android:id="@+id/spinner1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="24dp"
        android:layout_marginTop="24dp" />
```

XML 布局资源代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <TextView
        android:id="@+id/tv"
        android:layout_width="200dp"
        android:layout_height="wrap_content"
        android:text="TextView"
        android:textSize="20sp"
        android:background="#ffff00"
        android:layout_marginTop="5dp"/>

    <CheckBox
        android:id="@+id/checkBox1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="修改" />

</LinearLayout>
```

Java 代码如下：

```
//1.新建数据
    List<String> data=new ArrayList<String>();
    data.add("唐三藏");
    data.add("悟空");
    data.add("八戒");
    data.add("沙僧");
    data.add("妖怪");
//2.适配器,匹配数据
    //param1:上下文应用
    //param2:每个选项的资源文件
    //param3:textview的 id
    //param4:数据
    ArrayAdapter<String> adapter=new ArrayAdapter<String>(this, R.layout.item,
```

```
            R.id.tv, data);
//3.控件连接数据
    Spinner spinner= (Spinner) findViewById(R.id.spinner1);
    spinner.setAdapter(adapter);
    spinner.setPrompt("西游记人物");
```

效果如图 5-8 所示。



图 5-8　下拉框 1 效果图

5．ListView 的使用方法

(1) ListView 概述

ListView 是 Android 软件开发中非常重要的组件之一，可以绘制出漂亮的列表。

一个 ListView 通常有两个职责：①将数据填充到布局；②处理用户的选择单击等操作。

一个 ListView 的创建需要三个元素：①ListView 中每一列的 View；②填入 View 的数据或者图片等；③连接数据与 ListView 的适配器。

也就是说，要使用 ListView，首先要了解什么是适配器。适配器是一个连接数据和 AdapterView(ListView 就是一个典型的 AdapterView)的桥梁，通过它能有效地实现数据与 AdapterView 的分离设置，使 AdapterView 与数据的绑定更加简便，修改更加方便。

Android 中提供了很多的 Adapter，表 5-1 列出了常用的几个。

表 5-1　常用适配器

Adapter	含　义
ArrayAdapter<T>	用来绑定一个数组，支持泛型操作
SimpleAdapter	用来绑定在 XML 中定义的控件对应的数据

SimpleCursorAdapter	用来绑定游标得到的数据
BaseAdapter	通用的基础适配器

适配器还有很多，要注意的是，各种 Adapter 转换的方式和能力不一样。用 ArrayAdapter 可以实现简单的 ListView 的数据绑定。默认情况下，ArrayAdapter 绑定每个对象的 toString 值到 layout 中预先定义的 TextView 控件上。ArrayAdapter 的使用非常简单。

① 在布局文件中加入一个 ListView 控件。

```
<ListView
    android:id="@+id/listView1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true">
</ListView>
```

② 添加一个子项布局文件 item. xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/tv"
        android:layout_width="fill_parent"
        android:layout_height="50dp"
        android:text="TextView"
        android:textSize="30sp"
        android:background="#ffff00"/>

</LinearLayout>
```

③ 在 Activity 中初始化。

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //创建 list
    ArrayList<String> data=new ArrayList<String>();
    data.add("清明节");
    data.add("劳动节");
    data.add("青年节");
    data.add("端午节");
    data.add("儿童节");
    //适配器
    ArrayAdapter<String> adapter=new ArrayAdapter<String>(this,
    R.layout.item, R.id.tv, data);
```

```
        //控件绑定适配器
        ListView listView= (ListView) findViewById(R.id.listView1);
        listView.setAdapter(adapter);
        //绑定监听器
        listView.setOnItemClickListener(listener);
    }

    OnItemClickListener listener= new OnItemClickListener() {

        @Override
        public void onItemClick(AdapterView<?> parent, View view,
        int position, long id) {
            String s=parent.getItemAtPosition(position).toString();
            Toast.makeText(MainActivity.this, s, 0).show();

        }
    };
```

使用的步骤分析如下：

① 定义一个数组存放 ListView 中 item 的内容。

② 通过实现 ArrayAdapter 的构造函数创建一个 ArrayAdapter 的对象。

③ 通过 ListView 的 setAdapter()方法绑定 ArrayAdapter。

效果如图 5-9 所示。



图 5-9　listview1 数据绑定效果图

(2) ListView 使用 SimpleAdapter

很多时候需要在列表中展示的不仅仅是文字，有时还需展示图片等，使用 SimpleAdapter 就可实现。SimpleAdapter 的使用也非常简单，功能也非常强大，可以自定义 ListView 中的 item 的内容，比如图片、多选框等。

① 在布局文件中增加一个 ListView 控件，与上个程序一样，然后增加一个子项布局文件 item. xml。

```
<TextView
        android:id="@+id/tv_id"
        android:layout_width="50dp"
        android:layout_height="wrap_content"
        android:text="Text"
        android:textSize="20sp"/>

<TextView
    android:id="@+id/tv_name"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:text="TextView"
    android:gravity="right"
    android:textSize="20sp"/>

<TextView
    android:id="@+id/tv_sex"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="TextView"
    android:gravity="right"
    android:textSize="20sp"/>
```

② 在 Activity 中初始化。

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //1.数据
    List<Map<String, String>> data=new ArrayList<Map<String,String>>();
    //第一行
    Map<String, String> line1=new HashMap<String, String>();
    line1.put("sid", "100");
    line1.put("sname", "王小二");
    line1.put("sex", "男");
    //第一行
    Map<String, String> line2=new HashMap<String, String>();
    line2.put("sid", "110");
    line2.put("sname", "张飞");
    line2.put("sex", "男");
    //第一行
    Map<String, String> line3=new HashMap<String, String>();
    line3.put("sid", "120");
    line3.put("sname", "赵云");
    line3.put("sex", "男");

    data.add(line1);
```

```
        data.add(line2);
        data.add(line3);

        //适配器
        SimpleAdapter adapter= new SimpleAdapter(this, data, R.layout.item, new String[]{"sid","sname","
        sex"}, new int[]{R.id.tv_id,R.id.tv_name,R.id.tv_sex});

        ListView listView= (ListView) findViewById(R.id.listView1);
        listView.setAdapter(adapter);
        listView.setOnItemClickListener(listener);
    }


    OnItemClickListener listener= new OnItemClickListener() {

        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position,long
            id) {
            Map line= (Map) parent.getItemAtPosition(position);
            String sid= line.get("sid").toString();
            String sname= line.get("sname").toString();
            String sex= line.get("sex").toString();
            Toast.makeText(MainActivity.this, sid+ "-->"+ sname+ "--"+ sex, 0).show();

        }
    };
```

使用 SimpleAdapter 的数据一般都是用 HashMap 构成的列表,列表的每一节对应 ListView 的每一行。通过 SimpleAdapter 的构造函数,将 HashMap 的每个键的数据映射到布局文件中对应控件上。这个布局文件一般根据自己的需要自己定义。梳理一下使用 SimpleAdapter 的步骤。

① 根据需要定义 ListView 每行所实现的布局。

② 定义一个 HashMap 构成的列表,将数据以键值对的方式存放在里面。

③ 构造 SimpleAdapter 对象。

④ 将 LsitView 绑定到 SimpleAdapter 上。

效果如图 5-10 所示。

(3) ListView 使用 BaseAdapter

在 ListView 的使用中,有时还需要在里面加入按钮等控件,实现单独的操作。也就是说,这个 ListView 不再只是展示数据,也不仅仅是这一行要处理用户的操作,而是里面的控件要获得用户的焦点。读者可以试试用 SimpleAdapter 添加一个按钮到 ListView 的条目中,会发现可以添加,但是却无法获得焦点,单击操作被 ListView 的 Item 所覆盖。这时最方便的方法就是使用灵活的适配器 BaseAdapter。

使用 BaseAdapter 必须写一个类继承它,同时 BaseAdapter 是一个抽象类,继承它必须实现它的方法。BaseAdapter 的灵活性就在于它要重写很多方法,如图 5-11 所示为继



图 5-10 lsitview2 数据绑定效果图

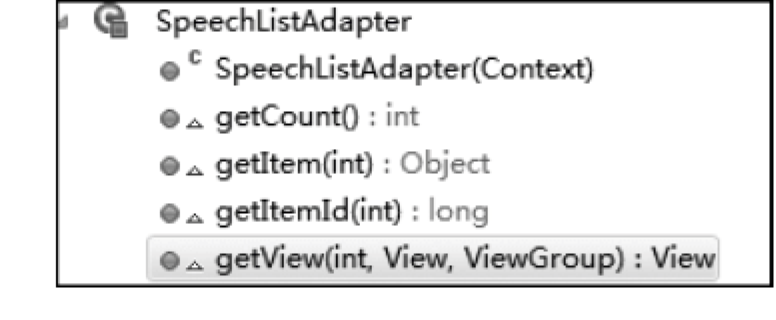


图 5-11 BaseAdapter 中的方法

承自 BaseAdapter 的 Speech-ListAdapter 所实现的方法，其中最重要的即为 getView()方法。这些方法都有什么作用呢？我们通过分析 ListView 的原理为读者解答。

当系统开始绘制 ListView 时，首先调用 getCount()方法，得到它的返回值，即 ListView 的长度。然后系统调用 getView()方法，根据这个长度逐一绘制 ListView 的每一行。如果让 getCount()返回 1，那么只显示一行。而 getItem()和 getItemId()则在需要处理和取得 Adapter 中的数据时调用。

布局文件和上一例类同，读者可以在光盘的工程目录中查看，这里只给出 Activity 类。

```
/**
    * 数据匹配视图的方法
    * param1:数据索引号,从 0开始
    * param2:一行控件视图
    * /
   @Override
   public View getView(int position, View convertView, ViewGroup parent) {
       //获取视图
       View view= inflater.inflate(R.layout.item, null);
       //分别获取视图中的三个控件
       TextView tv_id= (TextView) view.findViewById(R.id.tv_id);
       TextView tv_name= (TextView) view.findViewById(R.id.tv_name);
       TextView tv_sex= (TextView) view.findViewById(R.id.tv_sex);
       //设置数据
       Person person= persons.get(position);
```

```
            tv_id.setText(person.getId()+"");
            tv_name.setText(person.getName());
            tv_sex.setText(person.getSex());
            System.out.println(person.getId()+"------");
            return view;
        }
```

效果如图 5-12 所示。



图 5-12　listview4 数据绑定效果图

有时需要修改已经生成的列表，添加或者修改数据，notifyDataSetChanged()可以在修改适配器绑定的数组后，不用重新刷新 Activity，通知 Activity 更新 ListView。

Java 代码如下：

```
//滚动监听器
    OnScrollListener listener=new OnScrollListener() {
        //在滚动过程中,状态改变
        //第二个参数:状态
        @Override
        public void onScrollStateChanged(AbsListView view, int scrollState) {
            switch (scrollState) {
            //当前滚动状态为静止状态
            case OnScrollListener.SCROLL_STATE_IDLE:
                //并且 listview中最后一个用户可见的条目内容等于 listview适配器中最后一
                  个条目的内容
                //获取 listview中最后一个用户可见的条目的位置
                int position=view.getLastVisiblePosition();
                System.out.println("最后一个可见条目的位置:"+position);
                //获取适配器条目个数
                int totalCount=adapter.getCount();
                System.out.println("listview中条目个数:"+totalCount);
```

```
                if(position==(totalCount-1)){
                    //表示当前界面拖动到最下方
                    for (int i=-10; i<0; i++) {
                        data.add(i+"");
                    }
                    //通知适配器,数据已改变
                    adapter.notifyDataSetChanged();
                }
                break;

            default:
                break;
            }
        }

        @Override
        public void onScroll(AbsListView view, int firstVisibleItem,
                int visibleItemCount, int totalItemCount) {
            //TODO Auto-generated method stub

        }
    };
```

效果如图 5-13 所示。



图 5-13 listView 加载数据效果图

6. DatePicker 和 DatePickerDialog 的使用方法

DatePicker 继承 FrameLayout 类。

日期选择控件的主要功能是向用户提供包含年、月、日的日期数据并允许用户对其修改。如果要捕获用户修改日期选择控件的数据事件响应,需要为 DatePicker 添加一个 OnDateChangedListener 监听器。

重要方法如下。

- getDayOfMonth()：获取当前 Day。
- getMonth()：获取当前月(注意：返回数值为 0.11,需要自己+1 显示)。
- getYear()：获取当前年。
- updateDate(int year，int monthOfYear，int dayOfMonth)：更新日。
- init(int year，int monthOfYear，int dayOfMonth，DatePicker.OnDateChanged-Listener onDateChangedListener)。

参数如下。

- year：初始年。
- monthOfYear：初始月。
- dayOfMonth：初始日。
- onDateChangedListener：日期改变时通知用户的事件监听,可以为空(null)。

Java 代码如下：

```
@SuppressWarnings("deprecation")
    public void showDate(View view){
        //弹出对话框
        showDialog(1);
    }


    //1.日期设置监听器
    OnDateSetListener listener=new OnDateSetListener() {

        @Override
        public void onDateSet(DatePicker view, int year, int monthOfYear,
                int dayOfMonth) {
            System.out.println(year+"-->"+(monthOfYear+1)+"-->"+dayOfMonth);
        }
    };
    //2.设置弹出对话框
    protected android.app.Dialog onCreateDialog(int id) {
        switch (id) {
        case 1:
            //获取当前日期
            Calendar c=Calendar.getInstance();
            int year=c.get(Calendar.YEAR);
            int month=c.get(Calendar.MONTH);
            int day=c.get(Calendar.DAY_OF_MONTH);

            return new DatePickerDialog(this, listener, year, month, day);
        default:
            break;
        }
        return null;
    }
```

效果如图 5-14 所示。



图 5-14 日期组件效果图

7. AutoCompleteTextView 的使用方法

在 Android 中提供了智能输入框 AutoCompleteTextView。显示效果像 Google 搜索一样,当用户在搜索框里输入一些字符时(至少两个字符),会自动弹出一个下拉框提示类似的结果。

AutoCompleteTextView 常用属性。

- completionHint:设置出现在下拉菜单中的提示标题。
- completionThreshold:设置用户至少输入多少个字符才会显示提示。
- dropDownHorizontalOffset:下拉菜单于文本框之间的水平偏移,默认与文本框左对齐。
- dropDownHeight:下拉菜单的高度。
- dropDownWidth:下拉菜单的宽度。
- singleLine:单行显示。
- dropDownVerticalOffset:垂直偏移量。

重要方法如下。

- clearListSelection():清除选中的列表项。
- dismissDropDown():如果存在关闭下拉菜单。
- getAdapter():获取适配器。

XML 代码如下:

(1) 布局文件中设置 AutoCompleteTextView 控件。

```
<AutoCompleteTextView
        android:id="@+id/autoCompleteTextView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:ems="10"
        android:textSize="20sp">

        <requestFocus />
</AutoCompleteTextView>
```

(2) 设置子项布局文件 item. xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/tv"
        android:layout_width="fill_parent"
        android:layout_height="50dp"
        android:text="TextView"
        android:textSize="20sp"/>

</LinearLayout>
```

Java 代码如下：

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //1.数据
        List<String> data=new ArrayList<String>();
        data.add("tom");
        data.add("jack");
        data.add("jerry");
        data.add("jeep");
        //2.适配器
        ArrayAdapter<String> adapter=new ArrayAdapter<String>(this, R.layout.item, R.id.tv, data);
        //3.设置适配器
              AutoCompleteTextView textView = (AutoCompleteTextView) findViewById (R. id.
              autoCompleteTextView1);
         textView.setAdapter(adapter);

    }
```

效果如图 5-15 所示。



图 5-15 自动提示框效果图

8. ExpandableListView 的使用方法

ExpandableListView 用于显示组列表，显示效果如图 5-16 所示。



图 5-16 expandablelistview 显示效果图

组列表和列表在原理上是相似的，实现组列表的主要步骤有：①先建一个继承于 ExpandableListActivity 的 Activity；②有三个 XML 布局文件，main.xml 中有一个 ExpandableListView，代码如下。

(1) Activity 布局文件：activity_main.xml。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
```

```
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <ExpandableListView
        android:id="@+id/expandableListView1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true">
    </ExpandableListView>

</RelativeLayout>
```

（2）组项布局文件：group.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/tv_group"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        android:paddingLeft="30dp"
        android:textSize="20sp" />

</LinearLayout>
```

（3）子项布局文件：child.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/tv_child"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        android:paddingLeft="40dp"
        android:textSize="20sp"/>
```

```
</LinearLayout>
```

Java 代码如下：

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //数据
    //组
    List<Map<String, String>> groups= new ArrayList<Map<String,String>>();
    //group1
    Map<String, String> g1= new HashMap<String, String>();
    g1.put("group", "我的好友");
    //group2
    Map<String, String> g2= new HashMap<String, String>();
    g2.put("group", "陌生人");
    groups.add(g1);
    groups.add(g2);

    //子
    List<List<Map<String, String>>> childs= new ArrayList<List<Map<String,String>>>();
    //第一组子数据
    List<Map<String, String>> c1= new ArrayList<Map<String,String>>();
    //第一组子数据,第一行
    Map<String, String> c1_line1= new HashMap<String, String>();
    c1_line1.put("c1", "张三");
    //第一组子数据,第二行
    Map<String, String> c1_line2= new HashMap<String, String>();
    c1_line2.put("c1", "李四");
    //第一组子数据,第三行
    Map<String, String> c1_line3= new HashMap<String, String>();
    c1_line3.put("c1", "王五");

    c1.add(c1_line1);
    c1.add(c1_line2);
    c1.add(c1_line3);

    //第二组子数据
    List<Map<String, String>> c2= new ArrayList<Map<String,String>>();
    //第二组子数据,第一行
    Map<String, String> c2_line1= new HashMap<String, String>();
    c2_line1.put("c1", "刘德华");
    //第二组子数据,第二行
    Map<String, String> c2_line2= new HashMap<String, String>();
    c2_line2.put("c1", "张杰");
    //第二组子数据,第三行
```

```
        Map<String, String> c2_line3=new HashMap<String, String>();
        c2_line3.put("c1", "范冰冰");

        c2.add(c2_line1);
        c2.add(c2_line2);
        c2.add(c2_line3);

        //把第一组数据加入子数据
        childs.add(c1);
        childs.add(c2);

        //适配器
        //1.context
        //2.一级条目的数据
        //3.用来设置一级条目样式的布局文件
        //4.指定一级条目数据的 key
        //5.指定一级条目数据显示控件的 id
        //6.指定二级条目的数据
        //7.用来设置二级条目样式的布局文件
        //8.指定二级条目数据的 key
        //9.指定二级条目数据显示控件的 id
        SimpleExpandableListAdapter adapter=new SimpleExpandableListAdapter(this, groups, R.layout.group,
        new String[]{"group"}, new int[]{R.id.tv_group}, childs, R.layout.child, new String[]{"c1"}, new int
        []{R.id.tv_child});

        //绑定
        ExpandableListView listView=(ExpandableListView) findViewById(R.id.expandableListView1);
         listView.setAdapter(adapter);

    }
```

效果如图 5-17 所示。

5.2.2　项目界面相关代码设计

1. 登录界面

QQ 登录界面的布局文件为 activity_main. xml，效果如图 5-18 所示。

2. 好友界面

QQ 好友界面的布局文件为 layout_qqlist_expandable. xml，效果如图 5-19 所示。



图 5-17 二级列表效果图



图 5-18 QQ 登录界面效果图

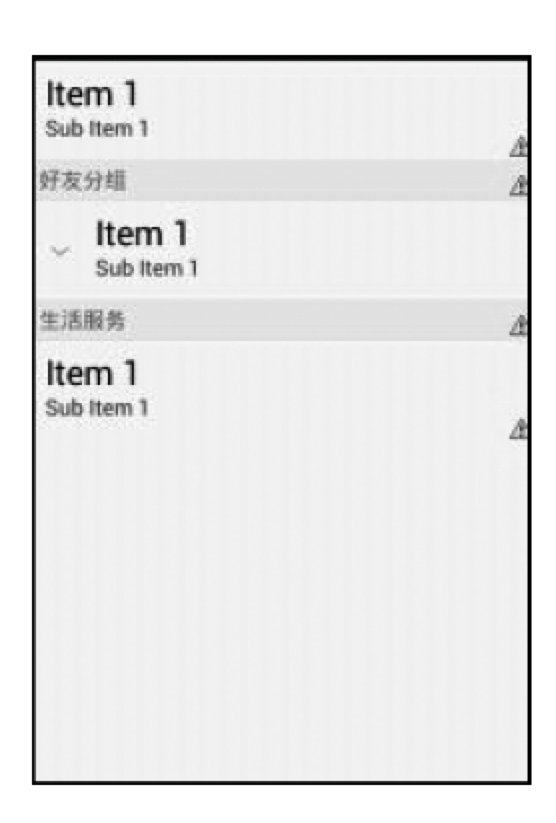


图 5-19 QQ 好友界面效果图

对应的 XML 代码如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#f1f1f1"
    tools:context=".MainActivity"
    android:scrollbars="vertical"
    android:scrollbarAlwaysDrawVerticalTrack="true">

    <ScrollView
        android:id="@+id/qqlist_scroll"
        android:layout_width="match_parent"
```

```
android:layout_height="wrap_content"
android:fadingEdge="none">

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <ListView
        android:id="@+id/qqlist_classify"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:divider="#000000"
        android:dividerHeight="0px"
        android:fadingEdge="none"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/qqlist_classify_text"
        android:text="好友分组"
        android:textSize="6pt"
        android:textColor="#666666"
        android:padding="4dip"/>

    <ExpandableListView
        android:id="@+id/qq_list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:divider="#888888"
        android:dividerHeight="0px"
        android:fadingEdge="none"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/qqlist_classify_text"
        android:text="生活服务"
        android:textSize="6pt"
        android:textColor="#666666"
        android:padding="4dip"/>

    <ListView
        android:id="@+id/qqlist_classify_down"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:divider="#000000"
        android:dividerHeight="0px"/>

</LinearLayout>
```

```
    </ScrollView>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="40dip"
        android:id="@+id/titleGroupView"
        android:orientation="horizontal"
        android:visibility="gone"
        android:background="@drawable/list_item_border">

        <ImageView
            android:id="@+id/title_groupImage"
            android:layout_width="match_parent"
            android:layout_height="16dip"
            android:layout_weight="1.8"
            android:layout_gravity="center"
            android:background="@drawable/todown" />

        <TextView
            android:id="@+id/title_groupName"
            android:layout_width="match_parent"
            android:layout_height="42dip"
            android:layout_weight="1"
            android:paddingLeft="15dip"
            android:paddingTop="11dip"
            android:textSize="7pt"
            android:text="fdg" />
        <TextView
            android:id="@+id/title_childCount"
            android:layout_width="match_parent"
            android:layout_height="42dip"
            android:layout_weight="1"
            android:gravity="right"
            android:paddingRight="10dip"
            android:paddingTop="11dip"
            android:text="ewsf"/>

    </LinearLayout>

</RelativeLayout>
```

在这个布局文件中包括 1 个 ExpandableListView 和 2 个 ListView，ExpandableListView 对应的子项布局文件为 layout_qqlist_group. xml 和 layout_qqlist_child. xml，2 个 ListView 对应的布局文件分别为 layout_list_item. xml 和 qqlist_classify_item. xml，如图 5-20 所示。

3. 聊天界面

QQ 聊天界面的布局文件为 activity_chat. xml，效果如图 5-21 所示。

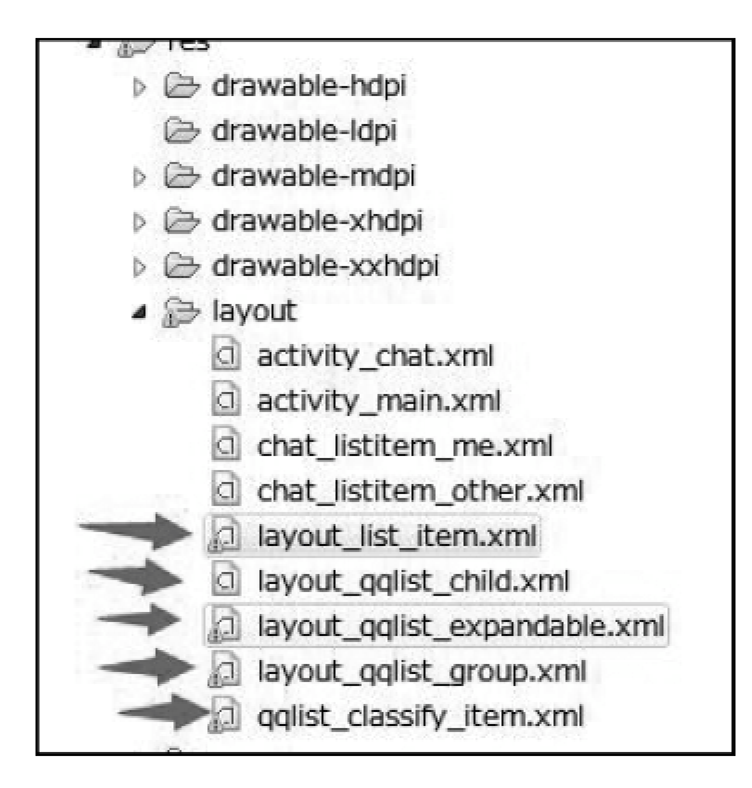


图 5-20　布局文件

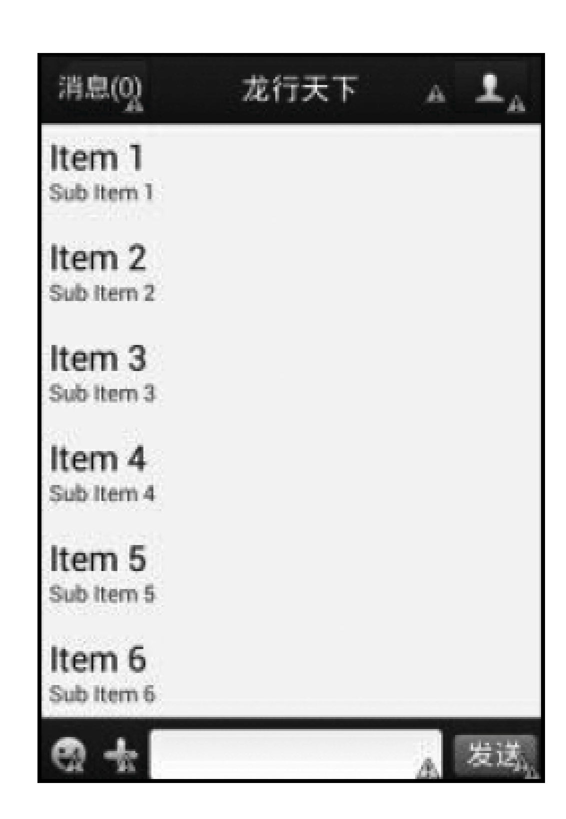


图 5-21　QQ 聊天界面效果图

XML 代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="44dip"
        android:id="@+id/chat_title"
        android:layout_alignParentTop="true"
        android:background="@drawable/chat_title_layer">
        <Button
            android:id="@+id/chat_msg_button"
            android:layout_width="match_parent"
            android:layout_height="36dip"
            android:layout_weight="1.9"
            android:layout_marginLeft="8dip"
            android:layout_marginTop="3dip"
            android:text="消息(0)"
            android:textColor="@android:color/white"
            android:textSize="7pt"
            android:background="@drawable/msg_button_back"/>
        <TextView
            android:id="@+id/chat_contact_name"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="龙行天下"
            android:textSize="8pt"
            android:textColor="@android:color/white"
            android:gravity="center"
            android:layout_gravity="center_vertical"/>
```

```
        <ImageButton
            android:id="@+id/chat_contact_button"
            android:layout_width="match_parent"
            android:layout_height="36dip"
            android:layout_weight="2"
            android:layout_marginRight="8dip"
            android:layout_marginTop="3dip"
            android:background="@drawable/chat_contact_back"/>

    </LinearLayout>

    <LinearLayout
        android:id="@+id/chat_bottom_linear"
        android:layout_width="match_parent"
        android:layout_height="42dip"
        android:background="@drawable/chat_title_layer"
        android:orientation="horizontal"
        android:layout_alignParentBottom="true"
        android:paddingTop="7dip"
        android:paddingBottom="3dip">

        <ImageButton
            android:id="@+id/chat_bottom_look"
            android:layout_width="match_parent"
            android:layout_height="26dip"
            android:layout_weight="3.5"
            android:layout_marginLeft="7dip"
            android:layout_marginTop="3dip"
            android:background="@drawable/chat_bottom_look"/>
        <ImageButton
            android:id="@+id/chat_bottom_add"
            android:layout_width="match_parent"
            android:layout_height="26dip"
            android:layout_weight="3.5"
            android:layout_marginLeft="7dip"
            android:layout_marginTop="3dip"
            android:background="@drawable/chat_bottom_add"/>
        <EditText
            android:id="@+id/chat_bottom_edittext"
            android:layout_width="match_parent"
            android:layout_height="32dip"
            android:layout_marginLeft="5dip"
            android:layout_marginRight="7dip"
            android:layout_weight="1.5"
            android:background="@drawable/edit_fillet_shape"/>
        <Button
            android:id="@+id/chat_bottom_sendbutton"
            android:layout_width="match_parent"
            android:layout_height="26dip"
```

```
            android:layout_weight="3.2"
            android:layout_marginRight="4dip"
            android:layout_marginBottom="3dip"
            android:background="@drawable/chat_button_fillet_shape"
            android:text="发送"
            android:textColor="@android:color/white"/>"

    </LinearLayout>
    <ListView
        android:id="@+id/chat_list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/chat_title"
        android:layout_above="@id/chat_bottom_linear"
        android:fadingEdge="none"
        android:background="#f0f0f0"
        android:divider="#aaaaaa"
        android:dividerHeight="0px">
    </ListView>

</RelativeLayout>
```

聊天时为了区分显示自己发送的消息和对方发过来的消息，在 ListView 中使用不同的布局子项。chat_listitem_me.xml（消息中的文字和头像右显示）和 chat_listitem_other.xml（消息中的文字和头像居左显示）。

5.3 项目功能的实现

通过代码优化 QQ 聊天程序的界面效果，该部分代码在项目的 src 目录中完成，如图 5-22 所示。



图 5-22　src 目录

（1）触摸事件 onTouchEvent，在登录页面（MainActivity.java）中选择用户时，会弹出下拉列表选择用户，用户单击列表之外部分时需要让下拉列表消失，这时需要重写 Activity 的 onTouchEvent。

```
//重写 onTouchEvent 方法,用于实现单击控件之外的部分使控件消失的功能
```

```
@Override
public boolean onTouchEvent(MotionEvent event) {
    //TODO Auto-generated method stub
    if(event.getAction()==MotionEvent.ACTION_DOWN && isVisible){
        int[] location=new int[2];
        //调用getLocationInWindow方法获得某一控件在窗口中左上角的横纵坐标
        loginList.getLocationInWindow(location);
        //获得在屏幕上单击的点的坐标
        int x=(int)event.getX();
        int y=(int)event.getY();
        if(x<location[0]||x>location[0]+loginList.getWidth()||
                y<location[1]||y>location[1]+loginList.getHeight()){
            isIndicatorUp=false;
            isVisible=false;

            listIndicatorButton.setBackgroundResource(R.drawable.
            indicator_down);
            loginList.setVisibility(View.GONE);   //让ListView列表消失,并且让游标向下指!

        }

    }

    return super.onTouchEvent(event);
}
```

(2) 登录用户对象,登录界面中每个用户都有头像、账号、密码属性,可以把这些属性集中起来看成一个用户对象UserInfo,Java代码如下:

```
public class UserInfo {

    public int userPhoto;
    public String userQQ=null;
    public String userPassword=null;
    public int deleteButtonRes;
    public UserInfo(int userPhoto, String userQQ, String userPassword, int deleteButtonRes) {
        super();
        this.userPhoto=userPhoto;
        this.userQQ=userQQ;
        this.deleteButtonRes=deleteButtonRes;
        this.userPassword=userPassword;
    }

}
```

(3) 联系人对象,好友页面中每个好友都有姓名、昵称、头像、所在分组,可以把这些属性集中起来看成一个联系人对象ContactsInfo,Java代码如下:

```
public class ContactsInfo {
    public String userName=null;
```

```
        public String userSign= null;
        public int userImage= 0;
        public String groupInfo= null;
        public ContactsInfo(String userName, String userSign, int userImage,
                String groupInfo) {
            super();
            this.userName= userName;
            this.userSign= userSign;
            this.userImage= userImage;
            this.groupInfo= groupInfo;
        }

    }
```

(4) 重新设置组项数据和子项数据的高度，列表展开和收缩会影响二级列表中原有子项和组项数据高度，所以要通过代码重新定义组项数据和子项数据的高度。Java 代码如下：

```
View exGroupListItem= exListView.getExpandableListAdapter().getGroupView(0,false, null, exListView);
exGroupListItem.setLayoutParams(new LayoutParams(LayoutParams.WRAP_
CONTENT, LayoutParams.WRAP_CONTENT));
exGroupListItem.measure(0, 0);
GROUP_HEIGHT= exGroupListItem.getMeasuredHeight();

View exChildListItem= exListView.getExpandableListAdapter().getChildView(0, 0, false, null, exListView);
exChildListItem.setLayoutParams(new LayoutParams(LayoutParams.WRAP_
CONTENT,LayoutParams.WRAP_CONTENT));
exChildListItem.measure(0, 0);
CHILD_HEIGHT= exChildListItem.getMeasuredHeight();

ViewGroup.LayoutParams   params= exListView.getLayoutParams();
height= groupData.size() * GROUP_HEIGHT- 2;
params.height= height;
exListView.setLayoutParams(params);

for(int i= 0; i< groupData.size() ;i++){
    groupData.get(i).put("location",i * GROUP_HEIGHT);
}
```

(5) 滑动到某一个 Group 改 GroupItem 就在屏幕顶端悬停的效果，需要使用 onTouchEvent 方法，Java 代码如下：

```
private boolean isChecking= false;

    private class LocationCheckThread extends Thread{

        @Override
        public void run() {
            //TODO Auto- generated method stub
```

```
super.run();
isChecking= true;
int[] location= new int[2];
int beforeY= 0;
int i;
ExpandableHandler mHandler= new ExpandableHandler(Looper.
getMainLooper());
while(true){
    //exListView.getLocationOnScreen(location);

    exListView.getLocationOnScreen(location);
    if(beforeY= = location[1]){                    //控件停止运动,线程关闭
        isChecking= false;
        return;
    }
    else{
        beforeY= location[1];
        for(i= groupData.size()- 1; i> = 0; i- - ){
            if((Boolean)groupData.get(i).get("expanded")&&
            (Integer)groupData.get(i).get("location")+ location[1]< = 24){

                Message msg= new Message();
                msg.arg1= childData.get(i).size();
                msg.arg2= i;
                msg.obj= groupData.get(i).get("groupName");
                msg.what= VISIBILITY_VISIBLE;
                mHandler.sendMessage(msg);
                break;
            }
        }
        if(i< 0){
            Message msg= new Message();
            msg.what= VISIBILITY_GONE;
            msg.obj= "";                    //必须加这一步,否则会有空指针异常
            mHandler.sendMessage(msg);
        }

    }

    try {
        this.sleep(80);                    //sleep的时间不能过短!!
    } catch (InterruptedException e) {
        //TODO Auto- generated catch block
        e.printStackTrace();
    }

}
```

```
        }

    }

    public class ExpandableHandler extends Handler{

        public ExpandableHandler(Looper looper){
            super(looper);
        }
        @Override
        public void handleMessage(Message msg) {
            //TODO Auto-generated method stub
            super.handleMessage(msg);
            int listNum=msg.arg1;
            int visibility=msg.what;
            int groupPos=msg.arg2;
            String groupName=msg.obj.toString();

            if(visibility==VISIBILITY_GONE){
                titleGroupView.setVisibility(View.GONE);

                return;
            }
            else{

                titleGroupView.setVisibility(View.VISIBLE);
                titleGroupName.setText(groupName);
                titleChildCount.setText(""+listNum);

                titleGroupView.setTag(groupPos);
                                //给这个View控件设置一个标签属性,用于存放groupPosition
                /**
                 * setText()中的参数不能使int型!!
                 */
            }

        }

    }
```

(6)聊天信息的显示：自己发的消息和接收的消息需要使用不同的布局文件，在自定义的数据适配器中需要判断，Java代码如下：

```
class ViewHolder{
    public ImageView imageView=null;
    public TextView textView=null;
```

```
}

@Override
public View getView(int position, View convertView, ViewGroup
parent) {
    //TODO Auto-generated method stub
    ViewHolder holder=null;
    int who=(Integer)chatList.get(position).get("person");

        convertView=LayoutInflater.from(context).inflate(
                layout[who==ME?0:1], null);
        holder=new ViewHolder();
        holder.imageView=(ImageView)convertView.findViewById(to[who*2+0]);
        holder.textView=(TextView)convertView.findViewById(to[who*2+1]);

    System.out.println(holder);
    System.out.println("WHYWHYWHYWHYW");
    System.out.println(holder.imageView);

        holder.imageView.setBackgroundResource((Integer)chatList.
        get(position).get(from[0]));

        holder.textView.setText(chatList.get(position).get(from[1]).
        toString());
    return convertView;
}
```

5.4 系统运行与效果测试

系统运行与效果测试如图 5-23～图 5-27 所示。



图 5-24 登录界面

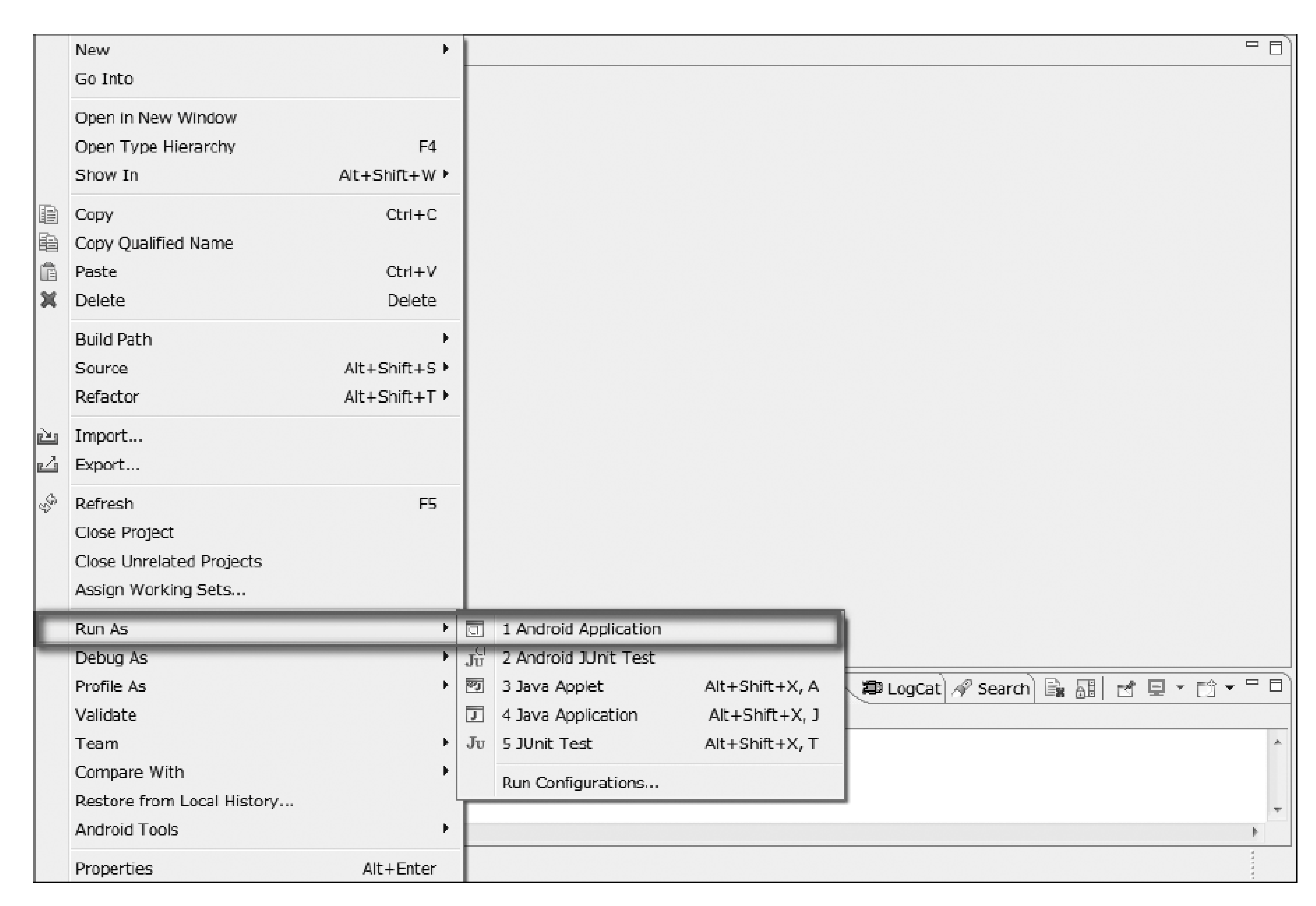


图 5-23　系统运行与效果测试



图 5-25 显示界面



图 5-26 我的好友



图 5-27　聊天

5.5　本章小结

通过本章项目练习，学习了 Android 的常用控件 RadioGroup 和 RadioButton、CheckBox、ProgressBar、Spinner、ListView、DatePicker 和 DatePickerDialog、AutoCompleteTextView、ExpandableListView，消息队列 Handler，触摸事件 onTouchEvent 等。

5.6　项目实践

(1) 尝试使用 Android 中其他控件。

(2) 理解 Handler 机制，掌握 Handler 应用场景。

(3) 根据现有的手机 QQ 版本优化应用程序。

第6章 短信智能管理器的设计及开发

本章的工作目标如下：

(1) 通过异步查询获取会话数据，对于会话进行删除和查询会话详情的功能。

(2) 对信息进行分类管理，并且对信息进行日期分隔显示。

(3) 学习群组模块。

(4) 完成程序运行与效果测试。

6.1 项目分析

本项目的学习和操作主要关注以下几点。

(1) 掌握碎片中 Tabhost 控件的基本使用。

(2) 掌握 Animation 动画效果。

(3) 学习 Android 中 SQLite 数据库的基本用法。

(4) 了解 Android 四大组件之一的 ContentProvider 的原理及使用。

(5) 综合以上几点，实现本章中的会话模块、文件夹模块和群组模块。

会话模块的实现如图 6-1 所示。

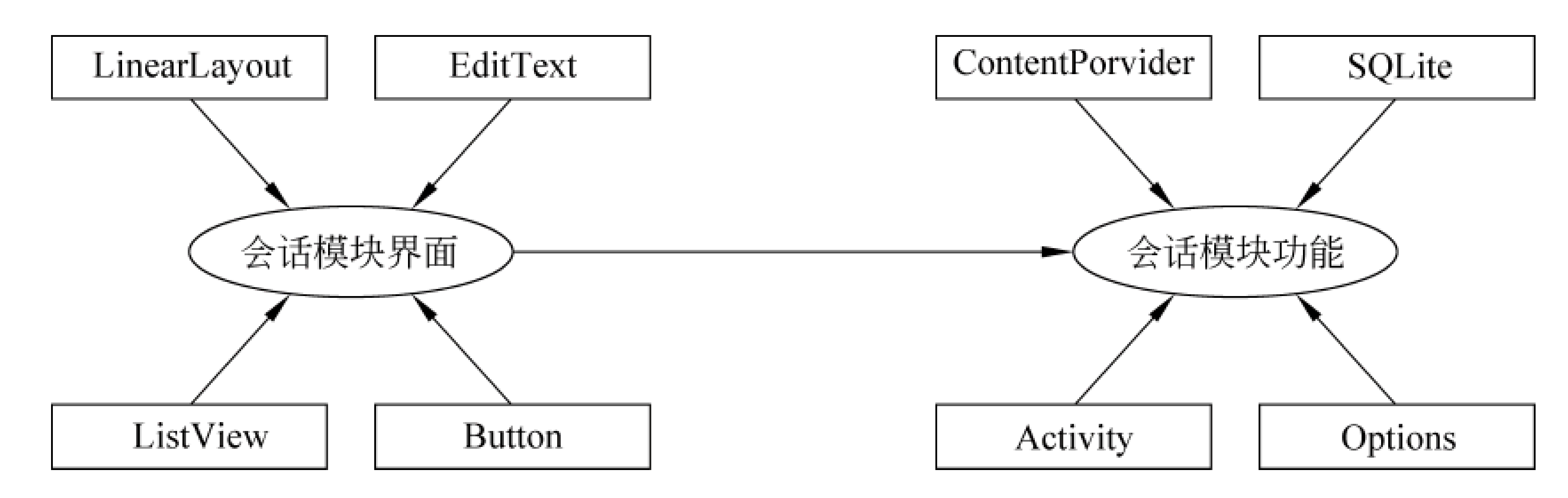


图 6-1 会话模块的实现

1. 会话模块界面的编写

会话模块共分为会话列表和会话详情两个界面。

会话列表界面由 ListView 和 Button 组成。最外层是 Orientation 为 Vertical 的 LinearLayout 其中由上自下依次为 Button 和 ListView。Button 的文字显示为“新建信息”。

会话详情界面由 ListView、Button、EditText 等组成。页面上方为会话的对象名称和返回键，中部是显示已发送信息的集合控件，下方为信息输入框和发送按钮。为了让输入框和发送按钮一直显示在界面的最下方，我们需要将 ListView 的高设置为 0 并且设置它的 Weight 属性为 1。

2. 会话模块功能的编写

首先通过 ContentProvider 查询数据库中的会话列表数据，然后使用 LsitView 和其对应的 Adapter 装载数据并显示出来。删除功能即用 ID 删除数据库中的信息后重复查询功能。会话详情的功能逻辑基本会话列表。

文件夹模块、群组模块界面和功能实现同会话模块。

6.2 项目界面设计

6.2.1 知识准备

1. Tabhost 控件

为了使界面更美观、运行效率更高，可以使用碎片化处理，主要包括 Tabhost 和 Fragment。本程序使用的是 Tabhost，共分为两个部分：页签和帧布局（各页面的显示）。在使用 Tabhost 时要注意控件的 ID 必须使用系统默认的，不可以更改。

```
<Tabhost
    android:id="@android:id/tabhost"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:oridentation="vertical" >

        <TabWidget
            android:id="@android:id/tabs"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:visibility="gone" >
        </TabWidget>

        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent" >
        </FrameLayout>
    </LinearLayout>
```

效果如图 6-2 所示。

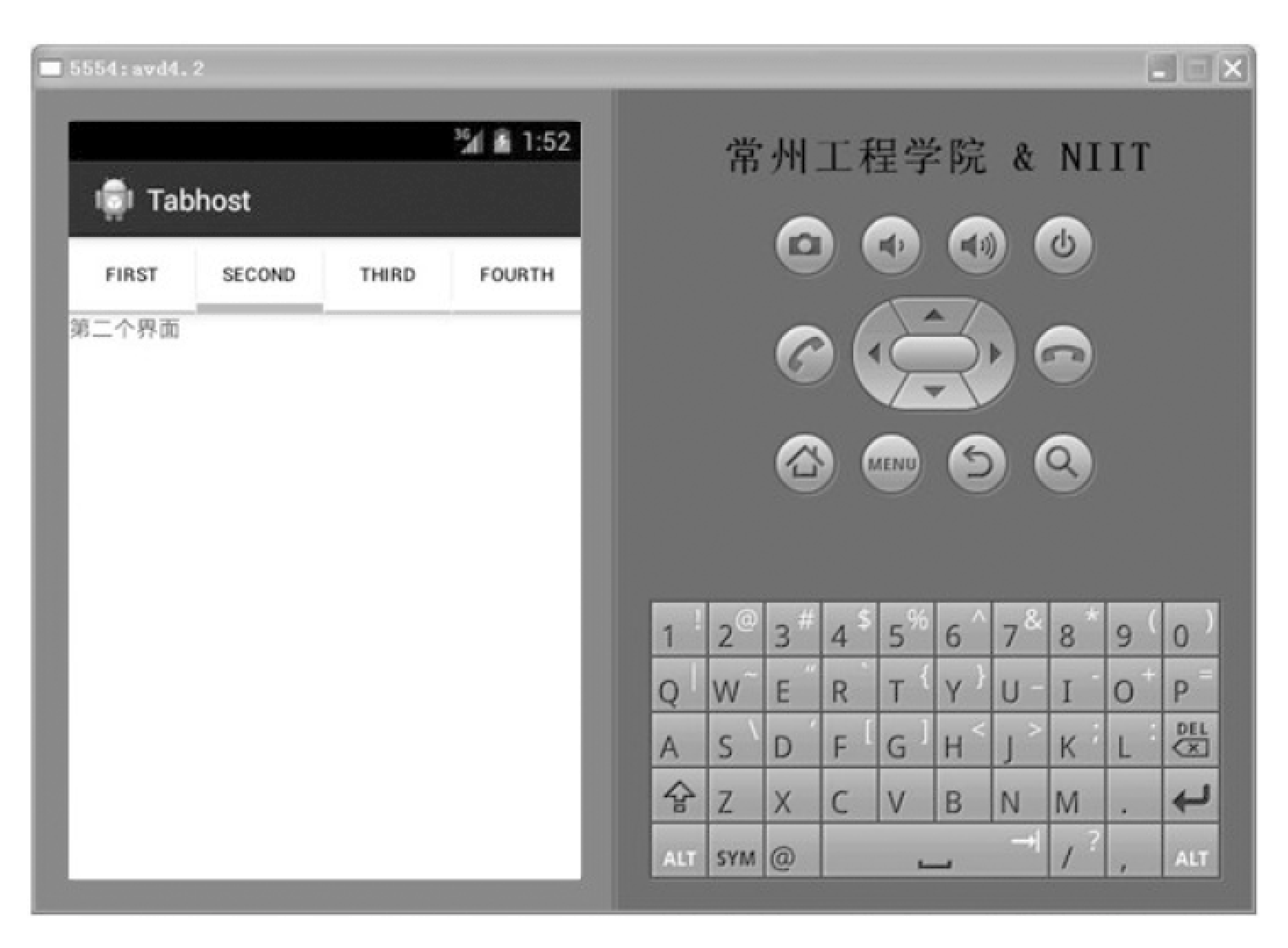


图 6-2　Tabhost 控件

2. View 实现动画效果

本项目使用 View 的动画效果显示当前所在的模块，此处只是简单地定义了此控件，其具体的大小和位置需要在 Java 代码中自定义。

```
<View
    android:id="@+id/slide_view"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/slide_background" />
```

效果如图 6-3 所示。



图 6-3　View 的动画效果图

6.2.2 项目界面相关代码设计

1. 页签界面的设计

上半部分为页签,下半部分为 FrameLayout。单击不同的页签将在 FrameLayout 中显示不同的界面。页签部分使用的并非默认的 TabWidget,而是 TextView 和 ImageView 的组合,通过 Java 代码实现与 Tabhost 的衔接,页签界面的设计如图 6-4 所示。

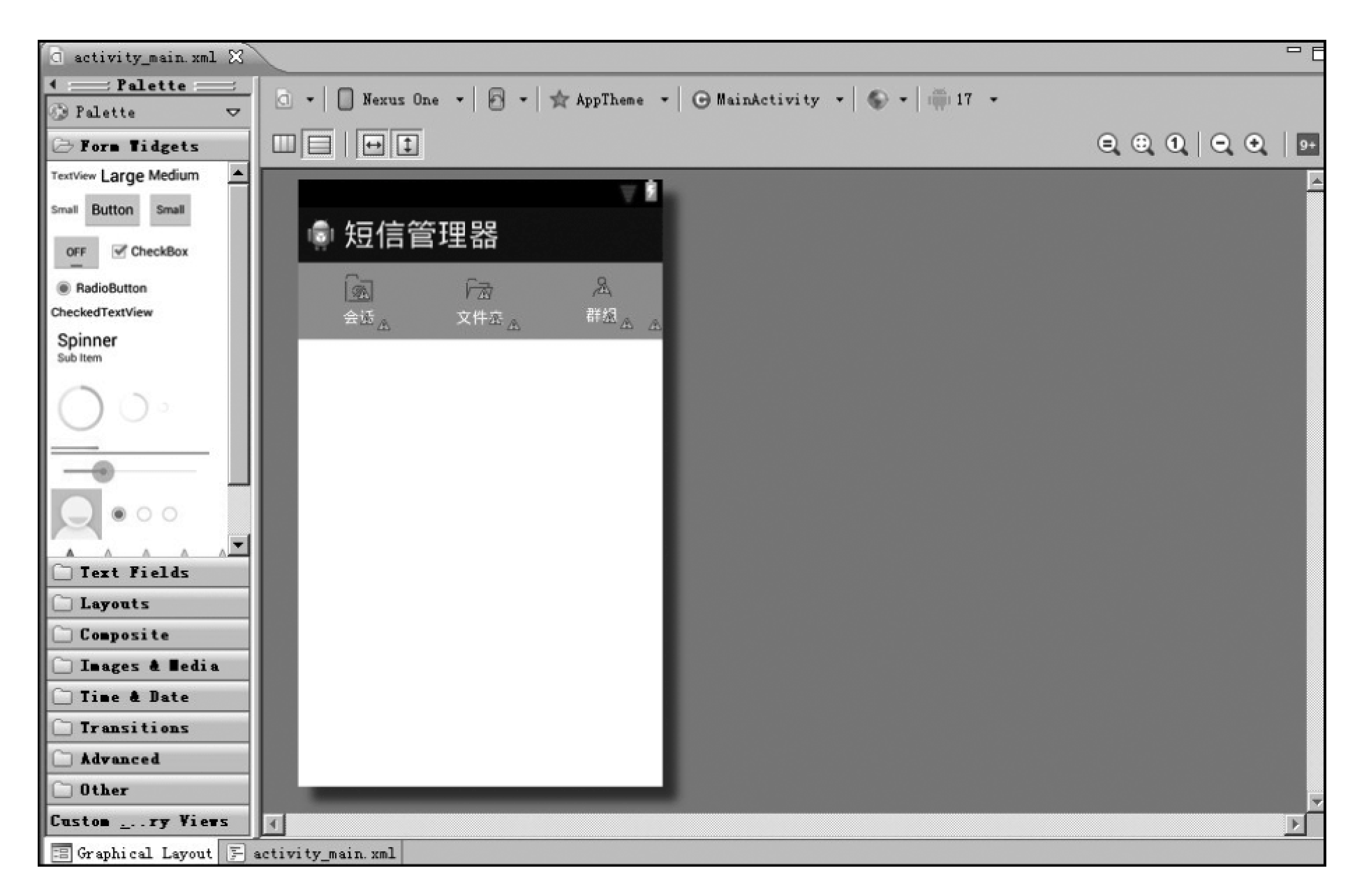


图 6-4 页签界面的设计

对应的 XML 代码如下:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">

    <Tabhost
        android:id="@android:id/tabhost"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:orientation="vertical">
```

```
<TabWidget
    android:id="@android:id/tabs"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:visibility="gone">
</TabWidget>

<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@android:color/darker_gray"
    android:paddingBottom="5dp"
    android:paddingTop="5dp">

    <View
        android:id="@+id/slide_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/slide_background"
        android:visibility="gone"/>

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">

        <RelativeLayout
            android:id="@+id/rl_conversation"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1">

            <LinearLayout
                android:id="@+id/ll_conversation"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_centerInParent="true"
                android:gravity="center_horizontal"
                android:orientation="vertical"
                android:paddingBottom="5dp"
                android:paddingLeft="15dp"
                android:paddingRight="15dp"
                android:paddingTop="5dp">

                <ImageView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:src="@drawable/tab_conversation" />
```

```
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="5dp"
                android:text="会话"
                android:textColor="@android:color/white" />
        </LinearLayout>
    </RelativeLayout>

    <RelativeLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1">

        <LinearLayout
            android:id="@+id/ll_folder"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:gravity="center_horizontal"
            android:orientation="vertical"
            android:paddingBottom="5dp"
            android:paddingLeft="15dp"
            android:paddingRight="15dp"
            android:paddingTop="5dp">

            <ImageView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:src="@drawable/tab_folder" />

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="5dp"
                android:text="文件夹"
                android:textColor="@android:color/white" />
        </LinearLayout>
    </RelativeLayout>

    <RelativeLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1">

        <LinearLayout
```

```
                    android:id="@+id/ll_group"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_centerInParent="true"
                    android:gravity="center_horizontal"
                    android:orientation="vertical"
                    android:paddingBottom="5dp"
                    android:paddingLeft="15dp"
                    android:paddingRight="15dp"
                    android:paddingTop="5dp">

                    <ImageView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_marginTop="2dp"
                        android:src="@drawable/tab_group" />

                    <TextView
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_marginTop="8dp"
                        android:text="群组"
                        android:textColor="@android:color/white" />
                </LinearLayout>
            </RelativeLayout>
        </LinearLayout>
    </RelativeLayout>

    <FrameLayout
        android:id="@android:id/tabcontent"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
    </FrameLayout>
  </LinearLayout>
</TabHost>

</LinearLayout>
```

2. 会话界面的设计

页签界面默认展示会话界面，或者在其他界面单击会话也可以进入会话界面。该界面有两种状态：默认状态和编辑状态。默认状态下显示新建信息按钮，单击进入新建信息界面，单击 ListView 进入会话详情界面。编辑状态下显示全选、取消和删除按钮，各有单击事件，会话界面的设计如图 6-5 所示。

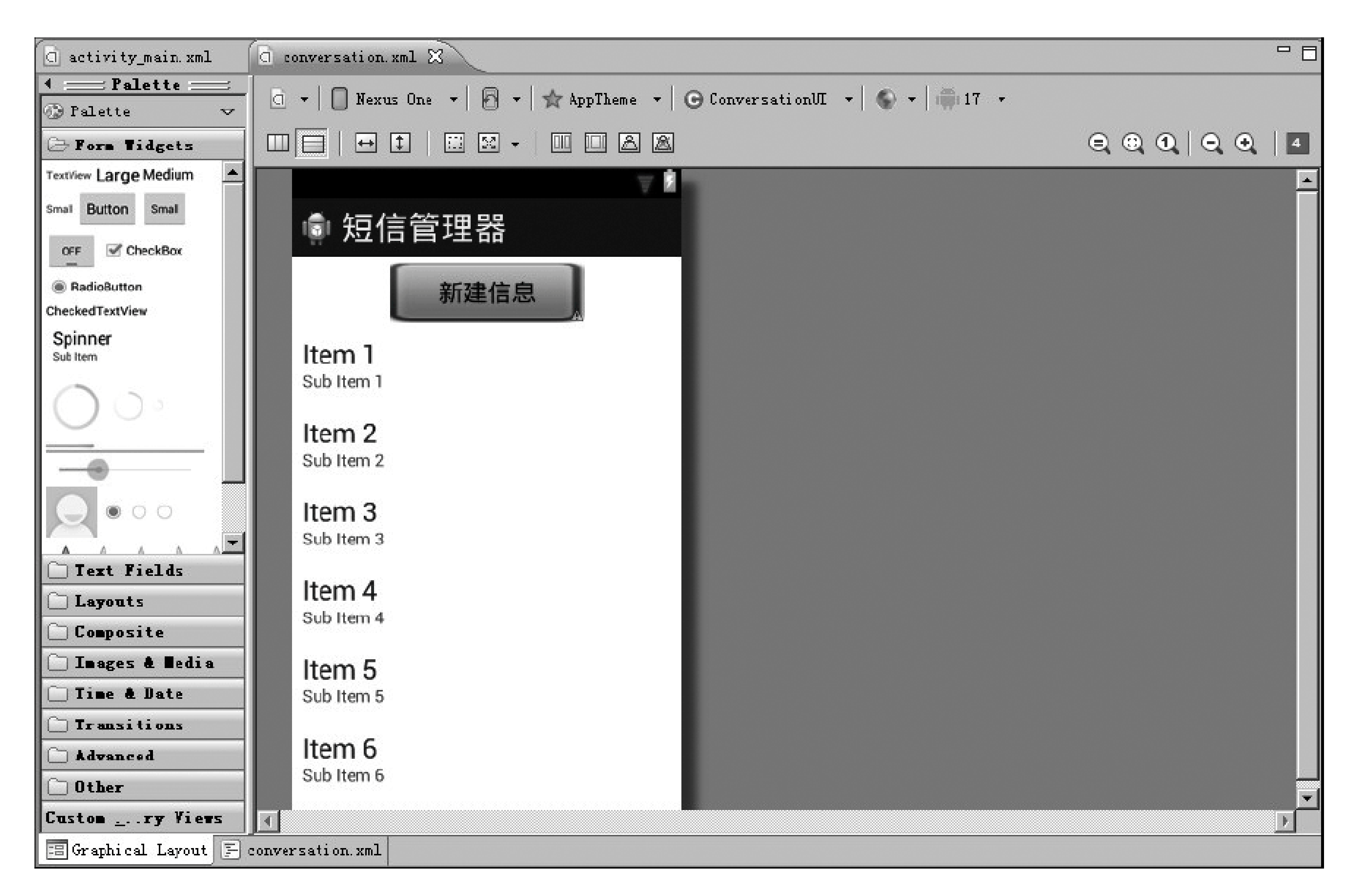


图 6-5 会话界面的设计

对应的 XML 代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingTop="5dp">

    <Button
        android:id="@+id/btn_conversation_new_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:background="@drawable/common_button_bg"
        android:paddingLeft="40dp"
        android:paddingRight="40dp"
        android:text="新建信息"
        android:textColor="@drawable/common_button_textcolor_selector"
        android:textSize="20sp" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp">

        <Button
```

```
            android:id="@+id/btn_conversation_select_all"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginRight="10dp"
            android:layout_weight="1"
            android:background="@drawable/common_button_bg"
            android:text="全选"
            android:textColor="@drawable/common_button_textcolor_selector"
            android:textSize="20sp"
            android:visibility="gone" />

        <Button
            android:id="@+id/btn_conversation_cancel_select"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:background="@drawable/common_button_bg"
            android:text="取消"
            android:textColor="@drawable/common_button_textcolor_selector"
            android:textSize="20sp"
            android:visibility="gone" />
    </LinearLayout>

    <ListView
        android:id="@+id/lv_conversation"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_marginTop="5dp"
        android:layout_weight="1">
    </ListView>

    <Button
        android:id="@+id/btn_conversation_delete_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="5dp"
        android:background="@drawable/common_button_bg"
        android:paddingLeft="40dp"
        android:paddingRight="40dp"
        android:text="删除信息"
        android:textColor="@drawable/common_button_textcolor_selector"
        android:textSize="20sp"
        android:visibility="gone" />

</LinearLayout>
```

3. 会话详情界面的设计

会话详情界面主要包括显示短信的 ListView、短信输入框、返回和发送短信按钮。

单击发送将短信发送出去并显示在 ListView 中，收到短信也可以及时地显示，单击返回则回到会话界面，会话详情界面的设计如图 6-6 所示。

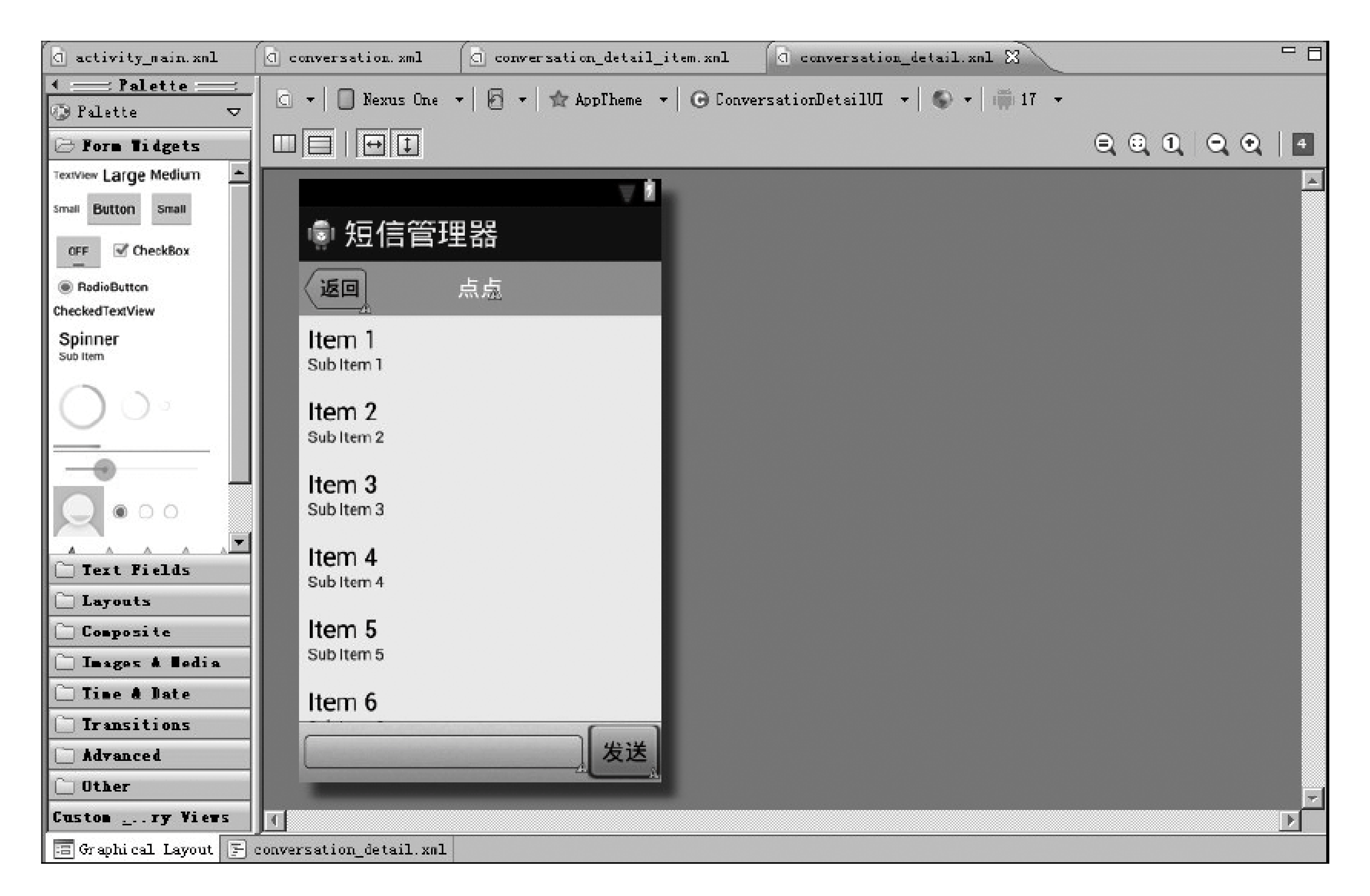


图 6-6　会话详情界面的设计

对应的 XML 代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:color/darker_gray">

        <Button
            android:id="@+id/btn_conversation_detail_back"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="@drawable/common_back_btn_bg"
            android:text="返回" />

        <TextView
            android:id="@+id/tv_conversation_detail_name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:text="点点"
```

```
            android:textColor="@android:color/white"
            android:textSize="20sp" />
    </RelativeLayout>

    <ListView
        android:id="@+id/lv_conversation_detail_sms"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:listSelector="@android:color/transparent"
        android:cacheColorHint="@android:color/transparent"
        android:background="@drawable/conversation_detail_content_bg"
        android:divider="@null"
        android:dividerHeight="0dp"
        android:transcriptMode="alwaysScroll">
    </ListView>

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/conversation_detail_footer_bg"
        android:gravity="center_vertical"
        android:orientation="horizontal">

        <EditText
            android:id="@+id/conversation_detail_content"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:maxLines="3"
            android:background="@drawable/common_deittext_bg" />

        <Button
            android:id="@+id/conversation_detail_send"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="@drawable/common_button_bg"
            android:text="发送"
            android:textColor="@drawable/common_button_textcolor_selector"
            android:textSize="20sp" />
    </LinearLayout>

</LinearLayout>
```

4. 新建信息页面的设计

新建信息页面主要包括联想查询输入框 AutoCompleteTextView，用来输入联系人号码、短信内容输入框、打开联系人列表的图片。单击联系人图片会隐式打开选择联系人

界面供选择，单击某个联系人返回该页面并将获取到的号码保存到 AutoComplete-TextView 中，新建信息页面的设计如图 6-7 所示。

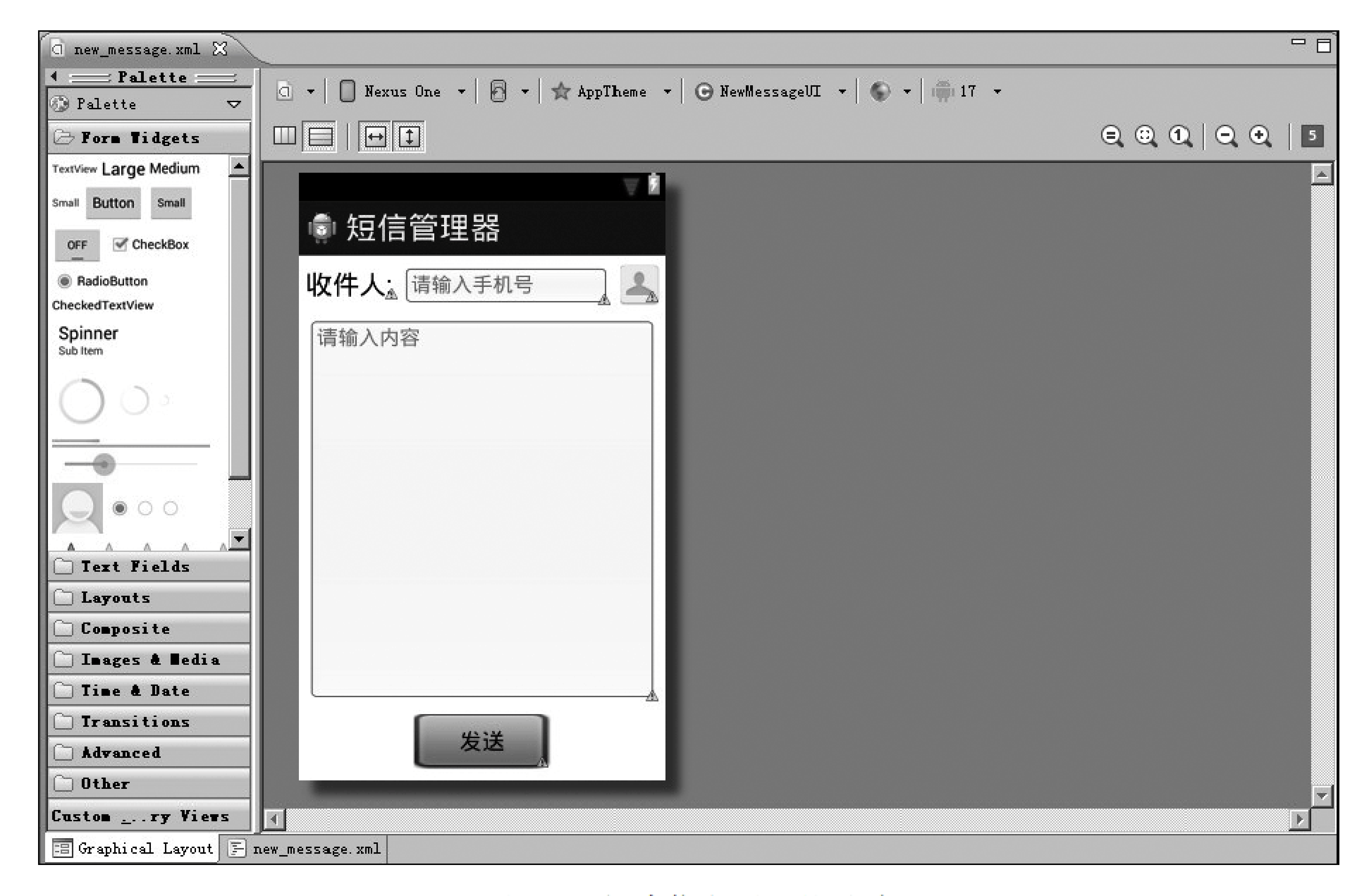


图 6-7　新建信息页面的设计

对应的 XML 代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:orientation="horizontal"
        android:padding="5dip">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="收件人："
            android:textColor="@android:color/black"
            android:textSize="23sp" />

        <AutoCompleteTextView
            android:id="@+id/actv_new_message_number"
```

```
            android:layout_width="0dip"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:singleLine="true"
            android:inputType="phone"
            android:hint="请输入手机号"
            android:textColor="@android:color/black"
            android:completionThreshold="1"
            android:background="@drawable/common_edittext_bg" />

        <ImageButton
            android:id="@+id/ib_new_message_select_contact"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="5dip"
            android:background="@drawable/select_contact_bg" />
    </LinearLayout>

    <EditText
        android:id="@+id/et_new_message_content"
        android:layout_width="fill_parent"
        android:layout_height="0dip"
        android:layout_marginLeft="5dip"
        android:layout_marginRight="5dip"
        android:gravity="left|top"
        android:layout_weight="1"
        android:hint="请输入内容"
        android:background="@drawable/common_edittext_bg" />

    <Button
        android:id="@+id/btn_new_message_send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:layout_margin="10dip"
        android:background="@drawable/common_button_bg"
        android:paddingLeft="40dip"
        android:paddingRight="40dip"
        android:text="发送"
        android:textColor="@drawable/common_button_textcolor_selector"
        android:textSize="20sp" />

</LinearLayout>
```

5．会话页面 ListView 的 item 设计

会话页面 ListView 的 item 设计如图 6-8 所示。



图 6-8　会话页面 ListView 的 item 设计

对应的 XML 代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:padding="5dp">

    <CheckBox
        android:id="@+id/cb_conversation_item"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:clickable="false"
        android:focusable="false"
        android:enabled="false"
        android:button="@drawable/common_checkbox_bg"
        android:visibility="gone"/>

    <ImageView
        android:id="@+id/iv_conversation_item_icon"
        android:layout_width="52dp"
        android:layout_height="52dp"
        android:layout_toRightOf="@id/cb_conversation_item" />

    <TextView
        android:id="@+id/tv_conversation_item_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
```

```
        android:layout_toRightOf="@id/iv_conversation_item_icon"
        android:text="张三(22)"
        android:textColor="@android:color/black"
        android:textSize="18sp" />

    <TextView
        android:id="@+id/tv_conversation_item_date"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:text="11:11"
        android:textColor="@android:color/darker_gray"
        android:textSize="14sp" />

    <TextView
        android:id="@+id/tv_conversation_item_body"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@id/tv_conversation_item_name"
        android:layout_below="@id/tv_conversation_item_name"
        android:layout_marginTop="5dp"
        android:singleLine="true"
        android:text="fdsfdsfds"
        android:textColor="@android:color/darker_gray"
        android:textSize="16sp" />

</RelativeLayout>
```

6. 文件夹页面 ListView 的 item 设计

文件夹页面 ListView 的 item 设计如图 6-9 所示。

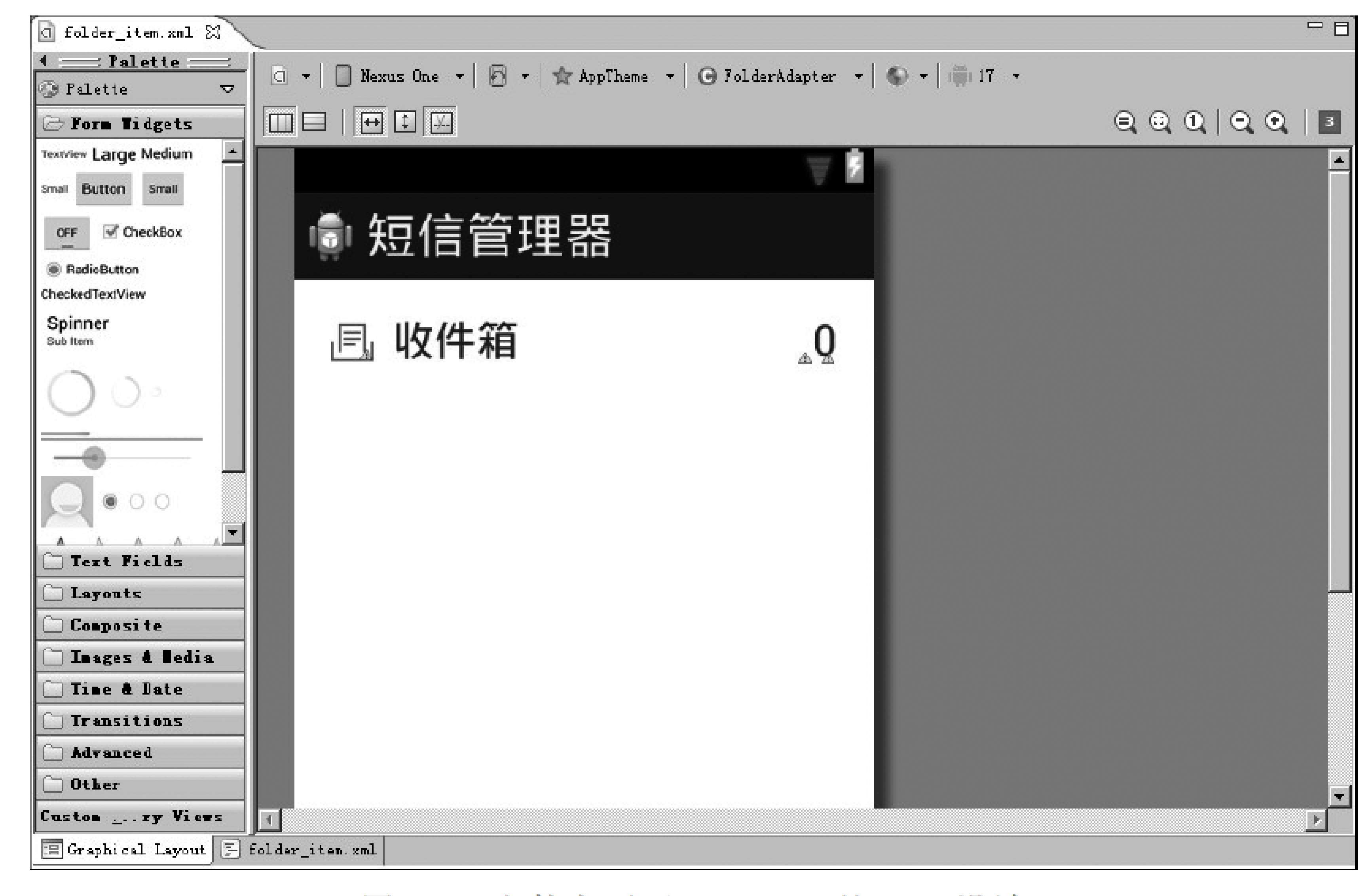


图 6-9 文件夹页面 ListView 的 item 设计

对应的 XML 代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:gravity="center_vertical"
    android:padding="20dp">

    <ImageView
        android:id="@+id/iv_folder_item_icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/a_f_draft"/>

    <TextView
        android:id="@+id/tv_folder_item_type"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:layout_marginLeft="10dp"
        android:text="收件箱"
        android:textSize="23sp"
        android:textColor="#660000"/>

    <TextView
        android:id="@+id/tv_folder_item_count"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0"
        android:textSize="23sp"
        android:textColor="#660000"/>

</LinearLayout>
```

7. 文件夹详情页面的设计

文件夹详情页面的设计如图 6-10 所示。

对应的 XML 代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
```

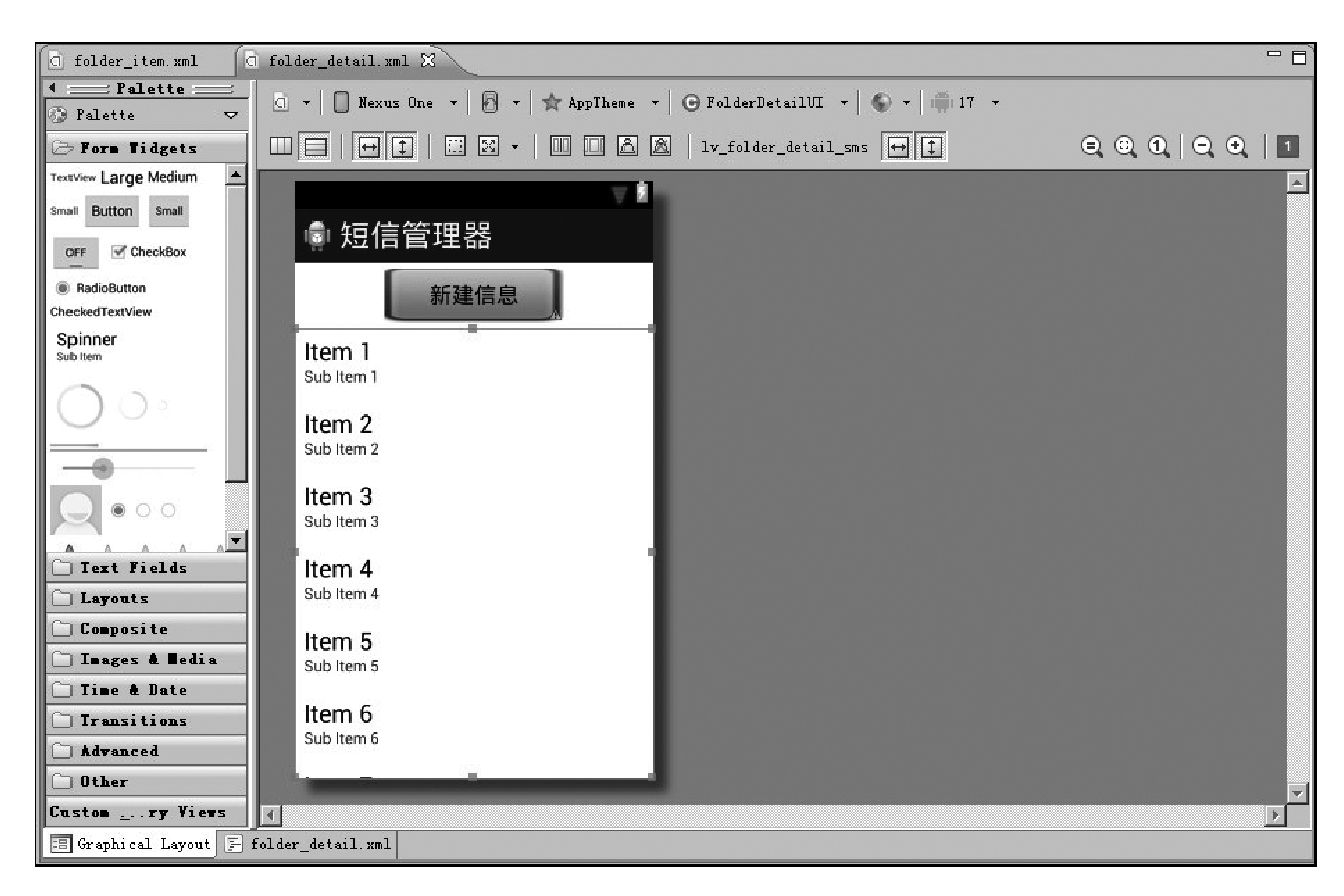


图 6-10 文件夹详情页面的设计

```
<Button
    android:id="@+id/btn_folder_detail_new_message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_marginTop="5dip"
    android:background="@drawable/common_button_bg"
    android:paddingLeft="40dip"
    android:paddingRight="40dip"
    android:text="新建信息"
    android:textColor="@drawable/common_button_textcolor_selector"
    android:textSize="20sp" />

<ListView
    android:id="@+id/lv_folder_detail_sms"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_marginTop="5dip">
</ListView>

</LinearLayout>
```

8. 新建群组界面的设计

新建群组界面为自定义的对话框界面，如图 6-11 所示。

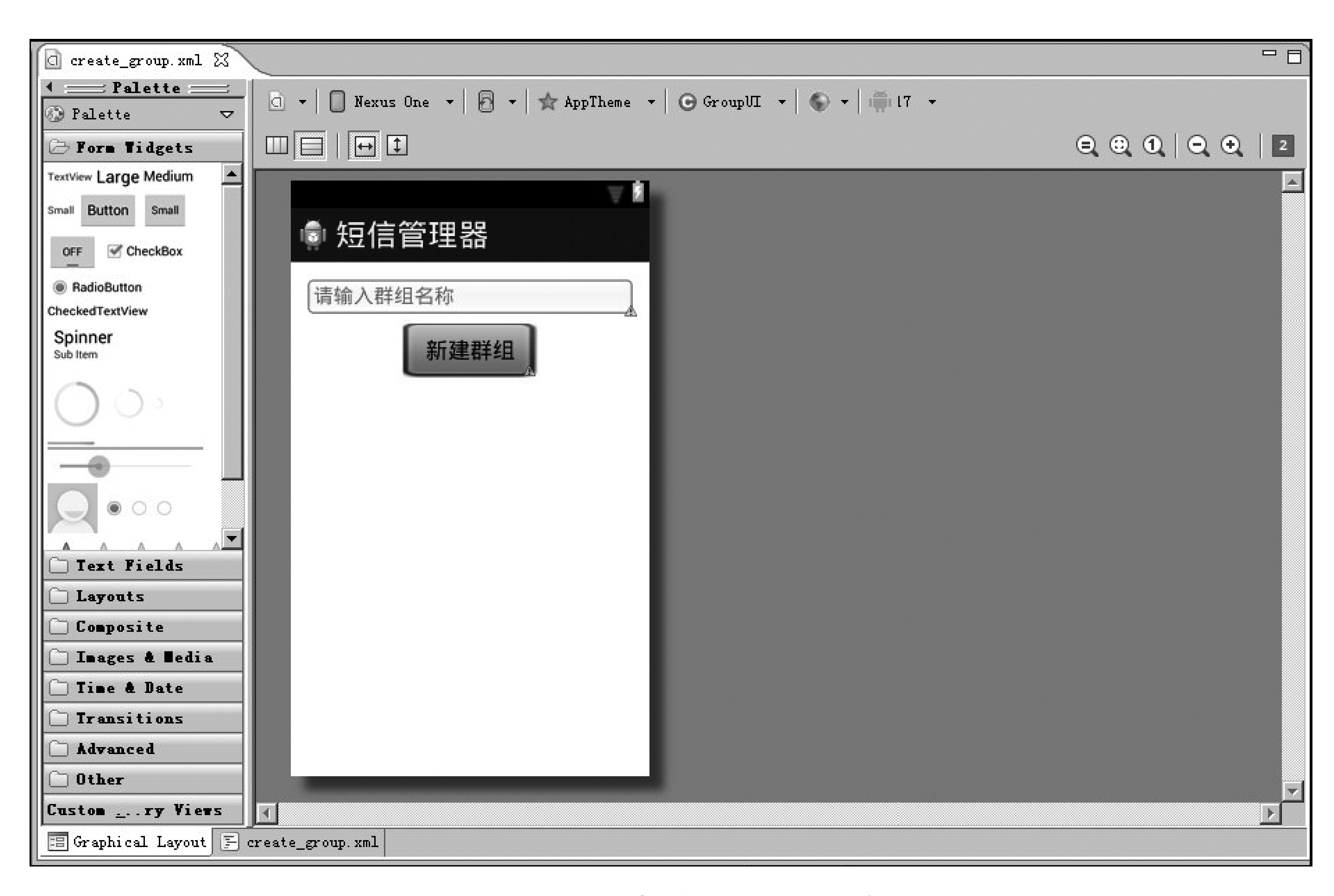


图 6-11　新建群组界面的设计

对应的 XML 代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="10dp"
    android:background="@android:color/white">

    <EditText
        android:id="@+id/et_create_group_name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="请输入群组名称"
        android:background="@drawable/common_edittext_bg"/>

    <Button
        android:id="@+id/btn_create_group"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/common_button_bg"
        android:text="新建群组"
        android:textColor="@drawable/common_button_textcolor_selector"
        android:textSize="20sp"
        android:paddingLeft="20dp"
```

```
        android:paddingRight="20dp"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="5dp"/>

</LinearLayout>
```

6.3 项目功能的实现

6.3.1 知识准备

1. SQLite 数据库

在 Android 平台上，集成了一个嵌入式关系型数据库 SQLite，SQLite 3 支持 NULL、INTEGER、REAL（浮点数字）、TEXT（字符串文本）和 BLOB（二进制对象）数据类型，虽然它支持的类型只有五种，但实际上 SQLite 3 也接受 varchar(n)、char(n)、decimal(p,s) 等数据类型，在运算或保存时会转换成对应的五种数据类型。SQLite 最大的特点是可以把各种类型的数据保存到任何字段中，而不需关心字段声明的数据类型。例如，可以在 INTEGER 类型的字段中存放字符串，或者在布尔型字段中存放浮点数，或者在字符型字段中存放日期型值。但有一种情况例外，定义为 INTEGER PRIMARY KEY 的字段只能存储 64 位整数，当向这种字段保存除整数以外的数据时，将会产生错误。另外，SQLite 在解析 CREATE TABLE 语句时，会忽略 CREATE TABLE 语句中跟在字段名后面的数据类型信息，如下面语句会忽略 name 字段的类型信息。

```
CREATE TABLE person (personid integer primary key autoincrement, name varchar(20))
```

SQLite 可以解析大部分标准 SQL 语句。

（1）查询语句：select * from 表名 where 条件子句 group by 分组字句 having...order by 排序子句。例如：

```
select * from person
    select * from person order by id desc
    select name from person group by name having count(*)>1
```

（2）分页 SQL 与 mySQL 类似，下面 SQL 语句获取 5 条记录，跳过前面 3 条记录。

```
select * from Account limit 5 offset 3
```

或者

```
select * from Account limit 3,5
```

（3）插入语句：insert into 表名(字段列表) values(值列表)。例如：

```
insert into person(name, age) values('niit',3)
```

（4）更新语句：update 表名 set 字段名=值 where 条件子句。例如：

```
update person set name='niit' where id=10
```

(5) 删除语句：delete from 表名 where 条件子句。例如：

```
delete from person where id=10
```

用户还可以使用 oncreate()和 onupgrade()方法实现数据库版本管理。当数据库被创建时会调用 oncreate()方法，并且该方法只会被调用一次。如果用户需要升级数据库，只需要将代码放入 onupgrade()方法中并改变版本号，当用户获取数据库实例时就会调用到 onupgrade()方法升级数据库。

getWritableDatabase()和 getReadableDatabase()方法都可以获取一个用于操作数据库的 SQLiteDatabase 实例。但 getWritableDatabase() 方法以读写方式打开数据库，一旦数据库的磁盘空间满了，数据库就只能读而不能写，倘若使用的是 getWritableDatabase()方法就会出错。getReadableDatabase()方法先以读写方式打开数据库，如果数据库的磁盘空间满了，就会打开失败，当打开失败后会继续尝试以只读方式打开数据库。

对于数据库的操作，用户可以选择使用 SQL 语句和系统 API 提供的类实现。

(1) SQL 语句。

exceSQL(预编译的 SQL 语句，参数数组)：此方法实现数据库的增、删、改。

rawQuery(预编译的 SQL 语句，参数数组)：此方法实现数据库的查询，返回值为 Curser 类型的游标结果集。

(2) 系统 API 查询方法。

增加：insert(表名，null，数据集合)，该方法返回 Long 型的添加行 ID，若返回－1 则表示添加失败。使用该方法需要创建数据集合，具体方法如下：

```
ContentValues values=new ContentValues();
Values.put(key,value);
```

删除：delete(表名，选择条件(参数用？表示)，参数数组)，该方法的返回值是 int 型的数值，表示影响的行数。

修改：update(表名，values，选择条件(参数用？表示)，参数数组)，该方法的返回值同 delete。

查询：query(表名，列名数组，查询条件，参数数组，分组条件，分组查询条件，排序条件)，返回值为 Curser 类型的游标结果集。

获取数据库的实例：

```
public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String name="niit";        //数据库名称
        private static final int version=1;         //数据库版本
            …
}
public class HelloActivity extends Activity {
    @Override public void onCreate(Bundle savedInstanceState) {
```

```
        ...
        Button button= (Button) this.findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v) {
            DatabaseHelper databaseHelper= new DatabaseHelper(HelloActivity.this);
            SQLiteDatabase db= databaseHelper.getWritableDatabase();
            db.execSQL("insert into person(name, age) values(?,?)", new Object[]{"niit", 4});
            db.close();
        }});
    }
}
```

效果如图 6-12 所示。



图 6-12 数据库的创建

2. ContentProvider

(1) ContentProvider 类使用介绍

当应用继承 ContentProvider 类，并重写该类用于提供数据和存储数据的方法时，就可以向其他应用共享其数据。之前学习过文件的操作模式，通过指定文件的操作模式为 Context. MODE_WORLD_READABLE 或 Context. MODE_WORLD_WRITEABLE 同样可以对外共享数据，但数据的访问方式会因数据存储的方式而不同，如采用 xml 文件对外共享数据，需要进行 xml 解析读写数据；采用 sharedpreferences 共享数据，需要使用 sharedpreferences API 读写数据。而使用 ContentProvider 共享数据的优点是统一数据访问方式。

通过 ContentProvider 对外共享数据的步骤如下：

① 继承 ContentProvider 并重写以下方法。

```
public class PersonContentProvider extends ContentProvider{
    public boolean onCreate()
    public Uri insert(Uri uri, ContentValues values)
    public int delete(Uri uri, String selection, String[] selectionArgs)
    public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)
    public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String
    sortOrder)
    public String getType(Uri uri)}
```

② 在 AndroidManifest.xml 中使用<provider>对该 ContentProvider 进行配置，为了能让其他应用找到该 ContentProvider，ContentProvider 采用了 authorities（主机名/域名）对它进行唯一标识，可以把 ContentProvider 看作一个网站，authorities 就是它的域名。

```
<manifest ...>
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <provider android:name=".PersonContentProvider" android:authorities=
        "cn.niit.providers.personprovider"/>
    </application>
</manifest>
```

（2）UriMatcher 类使用介绍

Uri 代表要操作的数据，所以通常需要解析 Uri，并从 Uri 中获取数据。Android 系统提供了两个用于操作 Uri 的工具类，分别为 UriMatcher 和 ContentUris。掌握它们的使用，会便于我们的开发工作。

UriMatcher 类用于匹配 Uri，它的用法如下：

① 统一注册需要匹配 Uri 的路径。

```
//常量 UriMatcher.NO_MATCH 表示不匹配任何路径的返回码
UriMatcher sMatcher=new UriMatcher(UriMatcher.NO_MATCH);
//如果 match()方法匹配 content://cn.niit.provider.personprovider/person 路径,返回匹配码为 1
sMatcher.addURI("cn.niit.provider.personprovider", "person", 1);
//添加需要匹配 uri,如果匹配就会返回匹配码
//如果 match()方法匹配 content://cn.niit.provider.personprovider/person/230 路径,返回匹配码为 2
sMatcher.addURI("cn.niit.provider.personprovider", "person/#", 2);
//#号为通配符
switch (sMatcher.match(Uri.parse("content://cn.niit.provider.personprovider/person/10"))) {
  case 1
    break;
  case 2
    break;
  default://不匹配
    break;
}
```

② 注册完需要匹配的 Uri 后，就可以使用 sMatcher.match(uri)方法对输入的 Uri 进行匹配，如果匹配就返回匹配码，匹配码是调用 addURI()方法传入的第三个参数，假

设匹配 content://cn. niit. provider. personprovider/person 路径,返回的匹配码为 1。

(3) ContentUris 类使用介绍

ContentUris 类用于获取 Uri 路径后面的 ID 部分,它有两个比较实用的方法。

① withAppendedId(uri, id)用于为路径加上 ID 部分。

```
Uri uri=Uri.parse("content://cn.niit.provider.personprovider/person")
Uri resultUri=ContentUris.withAppendedId(uri, 10);
//生成后的 Uri 为: content://cn.niit.provider.personprovider/person/10
```

② parseId(uri)方法用于从路径中获取 ID 部分。

```
Uri uri=Uri.parse("content://cn.niit.provider.personprovider/person/10")
long personid=ContentUris.parseId(uri);      //获取的结果为 10
```

(4) 使用 ContentProvider 共享数据

ContentProvider 类主要方法的作用如下:

① public boolean onCreate()。该方法在 ContentProvider 创建后就会被调用,Android 开机后,ContentProvider 在其他应用第一次访问它时才会被创建。

② public Uri insert(Uri uri, ContentValues values)。该方法用于供外部应用向 ContentProvider 添加数据。

③ public int delete(Uri uri, String selection, String[] selectionArgs)。该方法用于供外部应用从 ContentProvider 中删除数据。

④ public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)。该方法用于供外部应用更新 ContentProvider 中的数据。

⑤ public Cursor query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)。该方法用于供外部应用从 ContentProvider 中获取数据。

⑥ public String getType(Uri uri)。该方法用于返回当前 Url 所代表数据的 MIME 类型。如果操作的数据属于集合类型,那么 MIME 类型字符串应该以 vnd. android. cursor. dir/开头,例如,要得到所有 person 记录的 Uri 为 content://cn. niit. provider. personprovider/person,那么返回的 MIME 类型字符串应该为 vnd. android. cursor. dir/person。如果要操作的数据属于非集合类型数据,那么 MIME 类型字符串应该以 vnd. android. cursor. item/开头,例如,得到 ID 为 10 的 person 记录,Uri 为 content://cn. niit. provider. personprovider/person/10,那么返回的 MIME 类型字符串应该为 vnd. android. cursor. item/person。

效果如图 6-13 所示。

6.3.2 项目功能相关代码设计

1. 会话页面逻辑代码

会话页面的设计需要初始化页面、找到布局中的控件并且绑定监听器(OnClickListener)和适配器(CursorAdapter)。



图 6-13　内容提供者

```
private void initView(){
    mMultiDeleteSet=new HashSet<Integer>();
    mListView=(ListView) findViewById(R.id.lv_conversation);
    btnNewMessage=(Button) findViewById(R.id.btn_conversation_new_message);
    btnSelectAll=(Button) findViewById(R.id.btn_conversation_select_all);
    btnCancelSelect=(Button) findViewById(R.id.btn_conversation_cancel_
    select);
    btnDeleteMessage=(Button) findViewById(R.id.btn_conversation_delete_
    message);

    btnNewMessage.setOnClickListener(this);
    btnSelectAll.setOnClickListener(this);
    btnCancelSelect.setOnClickListener(this);
    btnDeleteMessage.setOnClickListener(this);
    mListView.setOnItemClickListener(this);

    mAdapter=new ConversationAdapter(this, null);
    mListView.setAdapter(mAdapter);
}
```

本页面有两个状态：列表和编辑。不同的状态下显示的控件不同，并且单击ListView后所做的操作也不一样。

```
private int currentState=LIST_STATE;        //当前默认的状态为列表状态
currentState==EDIT_STATE
```

在监听的回调方法根据单击的ID进行相应的操作，新建信息跳转新建信息页面。全选按钮被单击时将适配器中的游标结果集中数据全部放入删除选项集合中，并把游标结果集复位。取消按钮被单击时将删除选项集合中的数据清空。单击删除信息按钮，弹出删除信息对话框。

```
public void onClick(View view) {
    switch (view.getId()) {
    case R.id.btn_conversation_new_message:  //新建信息
        Intent intent= new Intent(this,NewMessageUI.class);
        startActivity(intent);
        break;

    case R.id.btn_conversation_select_all:   //全选
        Cursor cursor= mAdapter.getCursor();
        cursor.moveToPosition(-1);           //将游标结果集移动到初始位置

        while(cursor.moveToNext()){
            mMultiDeleteSet.add(cursor.getInt(THREAD_ID_COLUMN_INDEX));
        }
        mAdapter.notifyDataSetChanged();     //刷新数据
        refrechState();
        break;

    case R.id.btn_conversation_cancel_select:      //取消选择
        mMultiDeleteSet.clear();
        mAdapter.notifyDataSetChanged();
        refrechState();
        break;

    case R.id.btn_conversation_delete_message:     //删除信息
        showConfirmDeleteDialog();
        break;
    }
}
```

调用工具类中的 CommonAsyncQuery 对象查询数据，此查询方法为异步查询，可以直接在主线程中使用。查询完毕后的游标结果集在传入的 Adapter 中可以直接拿到。

```
private void prepareData(){
    CommonAsyncQuery asyncQuery= new CommonAsyncQuery(getContentResolver());
    asyncQuery.startQuery(0, mAdapter, Sms.CONVERSATION_URI,
    projection, null, null, "date desc");
}
```

CursorAdapter 可以直接在构造中获得游标结果集。

```
public ConversationAdapter(Context context, Cursor c) {
    super(context, c);
    //TODO Auto-generated constructor stub
}
```

newView 方法中得到 item 的视图和其中需要使用的控件并返回 bindView。

```
public View newView(Context content, Cursor curser, ViewGroup parent)
```

在 bindView 方法中使用游标结果集对获取到的视图控件进行数据绑定。

```
public void bindView(View view, Context context, Cursor cursor)
```

创建 options 菜单，只会被调用一次。

```
@Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //TODO Auto-generated method stub
        menu.add(0, SEARCH_ID, 0, "搜索");
        menu.add(0, EDIT_ID, 0, "编辑");
        menu.add(0, CANCEL_EDIT_ID, 0, "取消编辑");
        return super.onCreateOptionsMenu(menu);
    }
```

当菜单要显示在屏幕上时回调，控制显示哪一个菜单。

```
@Override
public boolean onPrepareOptionsMenu(Menu menu)
```

当 Options 菜单被选中时回调。

```
@Override
    public boolean onOptionsItemSelected(MenuItem item)
```

单击删除信息后弹出自定义对话框，确认是否删除，单击 OK 按钮弹出删除进度条对话框并开启子线程删除短信，为确保能看到进度条效果，本程序中使用 sleep。

```
private void showConfirmDeleteDialog() {
    Builder builder=new Builder(this);
    builder.setIcon(android.R.drawable.ic_dialog_alert);    //设置图标
    builder.setTitle("删除");
    builder.setMessage("确认删除选中的会话吗");
    builder.setPositiveButton("OK",new DialogInterface.OnClickListener(){

        @Override
        public void onClick(DialogInterface arg0, int arg1) {
            //弹出进度对话框
            showDeleteProgressDialog();
            //开启子线程，真正删除短信，每删除一条短信，更新进度条
            new Thread(new DeleteRunnable()).start();
        }
    });
    builder.setNegativeButton("Cancel", null);
    builder.show();

}
```

2. 会话详情页面逻辑代码

设置页面标题，根据 intent 传过来的参数 address 查询出联系人的姓名，此处的查询方法因为需要反复用到，所以写成了工具类。如果该名联系人没有设置姓名则使用号码代替。

```
private void initTitle() {
    Intent intent= getIntent();
    thread_id= intent.getIntExtra("thread_id",-1);
    address= intent.getStringExtra("address");

    String contactName= Utils.getContactName(getContentResolver(),
    address);

    TextView tvName= (TextView) findViewById(R.id.tv_conversation_detail_name);
    if(TextUtils.isEmpty(contactName)){
        tvName.setText(address);
    }else{
        tvName.setText(contactName);
    }

}
```

此页面的数据需要通过上一个传过来的 ID 取得并且按照时间顺序倒叙显示。

```
private void prepareData() {
    CommonAsyncQuery asyncQuery= new CommonAsyncQuery(getContentResolver());
    asyncQuery.startQuery(0, mAdapter, Sms.SMS_URI, projection, "thread_id=?", new String[]{thread_id+ ""},
    "date");
}
```

操作数据库中的信息和联系人表需要使用 contentprovider，即需要知道系统提供的 uri。例如：

```
Sms.SMS_URI
public static final Uri SMS_URI= Uri.parse("content://sms/");
```

此方法在 CurserAdapter 的内容发生改变时回调，将 ListView 移动到最后一行。

```
@Override
    protected void onContentChanged() {
        super.onContentChanged();
        mListView.setSelection(mListView.getCount());
    }
```

单击事件回调，单击发送在判断完号码和内容后将短信发送出去并将输入框的内容置空。单击返回按钮调用 finish 销毁此页面。

```
@Override
    public void onClick(View view) {
        switch (view.getId()) {
        case R.id.conversation_detail_send:
            String content= etContent.getText().toString();
            if(TextUtils.isEmpty(content)){
                Toast.makeText(this, "短信内容不能为空", Toast.LENGTH_SHORT).
                show();
                break;
```

```
            }
            //发送短信
            Utils.sendMessage(this, address, content);
            etContent.setText("");
            break;

        case R.id.btn_conversation_detail_back:
            finish();
            break;

        }

    }
```

3. 新建信息页面逻辑代码

新建信息页面需设置自动提示文本框，singleLine 设置显示的行数 inputType 设置可输入的内容。

```
<AutoCompleteTextView
        android:id="@+id/actv_new_message_number"
        android:layout_width="0dip"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:singleLine="true"
        android:inputType="phone"
        android:hint="请输入手机号"
        android:textColor="@android:color/black"
        android:completionThreshold="1"
        android:background="@drawable/common_edittext_bg" />
```

设置内容查询提供者，当自动提示文本框开始过滤查询时回调。

```
@Override
    public Cursor runQueryOnBackgroundThread(CharSequence constraint) {

        //Phone.CONTENT_URI 默认是查出所有联系人的号码
        String selection="datal like ?";
        String selectionArgs[]={constraint+"% "};
        Cursor cursor=getContentResolver().query(Phone.CONTENT_URI, contact_projection, selection,
        selectionArgs, null);

        return cursor;
    }
```

当单击提示列表时，显示在输入框显示的内容。

```
@Override
    public CharSequence convertToString(Cursor cursor) {
        return cursor.getString(ADDRESS_COLUMN_INDEX);
```

```
    }
```

隐式开启选择联系人界面。

```
Intent intent=new Intent(Intent.ACTION_PICK);
    intent.setData(Contacts.CONTENT_URI);  //获取当前系统中隐式调用联系人界面所需的 uri
    startActivityForResult(intent, 100);   //100 为请求码,为了区分是哪个程序调用
```

从开启的 Activity 被关闭时的回调函数 requestCode 为开启界面时的请求码,resultCode 为被开启界面的返回码。当前联系人有号码时将号码显示在输入框内,若没有号码则弹出 Toast 提示。

```
@Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if(requestCode==100 && resultCode==Activity.RESULT_OK){
            Uri uri=data.getData();
            int contactID=Utils.getContactID(getContentResolver(), uri);
            if(contactID !=-1){
                //当前联系人有号码
                String contactAddress=Utils.getContactAddress
                (getContentResolver(), contactID);
                actvNumber.setText(contactAddress);
                etContent.requestFocus();//编辑短信文本框取得焦点
            }else{
                Toast.makeText(this, "当前联系人尚未添加手机号", Toast.LENGTH_
                SHORT).show();
            }
        }

        super.onActivityResult(requestCode, resultCode, data);
    }
```

4. 文件夹页面逻辑代码

查询各文件夹中信息的条目数。

```
for(int i=0; i<4; i++){
    countMap.put(i, 0);

    uri=Utils.getUriFromIndex(i);

    asyncQuery.startQuery(i, null, uri, new String[]{"count(*)"}, null, null, null);
}
```

当刷新完数据之后回调此方法,将 cursor 传给 mAdapter 并且通知 mAdapter 数据改变,刷新页面数据。

```
@Override
    public void onPostNotify(int token, Object cookie, Cursor cursor) {
        if(cursor !=null && cursor.moveToFirst()){
```

```
                countMap.put(token, cursor.getInt(0));
                mAdapter.notifyDataSetChanged();
            }

        }
```

单击 ListView 中的某一项跳转到文件夹详情页面。

```
@Override
    public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
        //TODO Auto-generated method stub
        Intent intent=new Intent(this,FolderDetailUI.class);
        intent.putExtra("index", arg2);
        startActivity(intent);
    }
```

5. 文件夹详情页面逻辑代码

当前页面需要使用日期对短信进行归类，并且日期应属于 ListView 的一部分，此时就需要复写 CursorAdapter 中的 getView 方法。如果当前的 position 在集合日期中可以取到值时返回 TextView，若取不到则返回短信的 item。

使用 Adapter 的 ListView 可以实现 item 的复用，如果将 TextView 的对象拿来显示短信 item 数据，就会发生异常导致系统程序崩溃。所以，当 ListView 滚动获取缓存中的 convertView 时需要对 convertView 进行判断，若 convertView 不存在或者类型是 TextView，就需要重新设置一个短信 item 对象。

```
@Override
    public View getView(int position, View convertView, ViewGroup parent) {
        //当 position 可以在日期集合中取到值时,返回的是 textview
        if(dateMap.containsKey(position)){
            //当前需要显示日期
            TextView tvDate=new TextView(FolderDetailUI.this);
            tvDate.setBackgroundResource(android.R.color.darker_gray);
            tvDate.setTextSize(20);
            tvDate.setTextColor(Color.WHITE);
            tvDate.setGravity(Gravity.CENTER);
            tvDate.setText(dateMap.get(position));
            return tvDate;

        }
        //返回短信的 item
        Cursor mCursor=mAdapter.getCursor();
        mCursor.moveToPosition(smsRealPositionMap.get(position));
            View v;
            //convertView instanceof TextView 当 listview滚动获取缓存中的
              convertView为 TextView类型时,重新设置一个 item对象
            if (convertView==null || convertView instanceof TextView) {
```

```
                v= newView(FolderDetailUI.this, mCursor, parent);
            } else {
                v= convertView;
            }
            bindView(v, FolderDetailUI.this, mCursor);
            return v;
    }
```

当 Adapter 更新之前回调此方法(用户做一些适配数据之前的准备操作),此方法中的操作用于将短信按日期归类显示在界面上。需要 dateMap 和 smsRealPositionMap 两个集合,判断该索引中的日期在 dateMap 中是否已经存在,若不存在就将该数据的索引和日期存入集合,并将 ListView 中的索引++ 。之后将 listview 中的索引和该条数据的真实索引存入 smsRealPositionMap 中,最后将游标复位到－1 方便以后对游标结果集的操作。

```
@Override
    public void onPreNotify(int token, Object cookie, Cursor cursor) {
        if(cursor != null && cursor.getCount()> 0){

            java.text.DateFormat dateFormat= DateFormat.getDateFormat(this);
            String strDate= null;
            int listViewIndex= 0;//listView中的索引
            while(cursor.moveToNext()){
                long date= cursor.getLong(DATE_COLUMN_INDEX);
                strDate= dateFormat.format(date);
                //判断当前短信日期是否存在集合中,不存在就存一份
                if(!dateMap.containsValue(strDate)){
                    dateMap.put(listViewIndex, strDate);
                    listViewIndex+ + ;
                }

                smsRealPositionMap.put(listViewIndex, cursor.getPosition());
                listViewIndex+ + ;
                //把当前短信的真实索引存放在 smsRealPositionMap 中
            }
            //把游标复位到- 1
            cursor.moveToPosition(- 1);

        }

    }
```

6. 群组页面逻辑代码

创建 Options 菜单的另一种方法是在 Menu 文件夹下创建 menu 的 xml 文件后在其 Java 代码中调用。

```
@Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.create_group, menu);      //将 xml 文件添加到菜单选项中
        return super.onCreateOptionsMenu(menu);
    }
```

当前页面继承 ListActivity 只需要 getListView 即可获得 ListView 对象。

```
ListView mListView= getListView();
```

弹出新建群组对话框。

```
private void showCreateGroupDialog()
```

在对话框中显示试图。

```
dialog.setView(view,0,0,0,0);
```

设置对话框所占整个屏幕的大小。

```
//获得对话框窗体的属性
    LayoutParams lp= dialog.getWindow().getAttributes();

    //整个屏幕的宽度
    lp.width= (int) (getWindowManager().getDefaultDisplay().getWidth() * 0.7);
    dialog.getWindow().setAttributes(lp);
```

将输入的名称作为参数掺入插入语句，插入群组的数据。

```
protected void createGroup(String groupName) {
        ContentValues values= new ContentValues();
        values.put("group_name", groupName);
        Uri uri= getContentResolver().insert(Sms.GROUPS_INSERT_URI, values);
        if(ContentUris.parseId(uri) != -1){
            Toast.makeText(this, "群组创建成功", Toast.LENGTH_SHORT).show();

        }
}
```

在内容提供者中加入以下代码，当数据库中的数据发生改变的同时，适配器刷新数据。

```
//通知此 uri 数据改变,数据改变了就会重新查询数据
    getContext().getContentResolver().notifyChange(Sms.GROUPS_QUERY_ALL_URI, null);
//给游标结果集设置一个通知的 uri
    cursor.setNotificationUri(getContext().getContentResolver(),Sms.GROUPS_
    QUERY_ALL_URI);
```

CursorAdapter 的自动刷新过程。

```
ContentResolver().notifyChange
->ChangeObserver.onChange
```

```
->CursorAdapter.onContentChanged
```

关键步骤：此方法会调用 cursor.reQuery 方法查询最新的结果。

```
->DataSetObserver.onChanged
->BaseAdapter.notifyDataSetChanged 开始刷新数据
```

6.4　系统运行与效果测试

系统运行与效果测试如图 6-14～图 6-19 所示。



图 6-14　全选模式



图 6-15　删除信息 Dialog



图 6-16　会话详情界面



图 6-17　已发送文件夹详情



图 6-18　新建群组



图 6-19　群组创建成功

6.5　本章小结

通过本章的学习，对 Android 的四大组件之一 ContentProvider 有了一个初步的认识，能够结合 SQLite 获取 Android 系统中的一些开放数据，并且对其进行一定程度的修改。界面方面，我们学习了碎片中的一个重要的部分 Tabhost。使用该控件可以很容易地做出漂亮的界面效果。

6.6 项目实践

(1) 把做好的 APP 安装到实验用手机,测试会话、文件夹和群组模块功能。

(2) 使用不同分辨率的模拟器测试效果。

(3) 使用不同版本的模拟器测试效果。

(4) 理解 ContentProvider 的实现原理及其用法。

(5) 掌握 SQLite 数据库的创建、更新和增、删、改、查等功能。

第7章 学生信息管理系统的设计及开发

本章的工作目标如下：

(1) 完成系统概要设计。

(2) 掌握 Android 网络通信原理。

(3) 掌握 Android Butterknife 框架。

(4) 实现 Splash 界面效果。

(5) 实现注册、登录功能。

(6) 实现学生信息管理功能。

(7) 实现密码修改功能。

7.1 项目分析

学生信息管理系统主要用于学校学生信息管理，总体任务是实现学生信息关系的系统化、科学化、规范化和自动化，其主要任务是用计算机对学生各种信息进行日常管理，如查询、修改、增加、删除等。

本项目主要包括以下几部分：Splash 界面设计，软件升级；注册、登录页面及功能；学生信息管理功能实现。

(1) 班级、课程的设置管理：教师可以根据本校具体情况在每学期开始时设置所需班级数量和人数。并设置本学期的课程。

(2) 权限管理：为了很好地保证系统的安全性，学校相关负责人可以设置不同类型人员的权限。

(3) 学生档案管理：学生档案的数量十分庞大，教务管理人员进行新生入学的档案录入及更改。其中包括学生个人信息的修改。

(4) 学生成绩管理：教务管理人员可以查询和修改学生的历年考试成绩，掌握学生学习情况，作为评定学生素质的数据依据。

(5) 密码修改。

本章结构框架如图 7-1 所示。

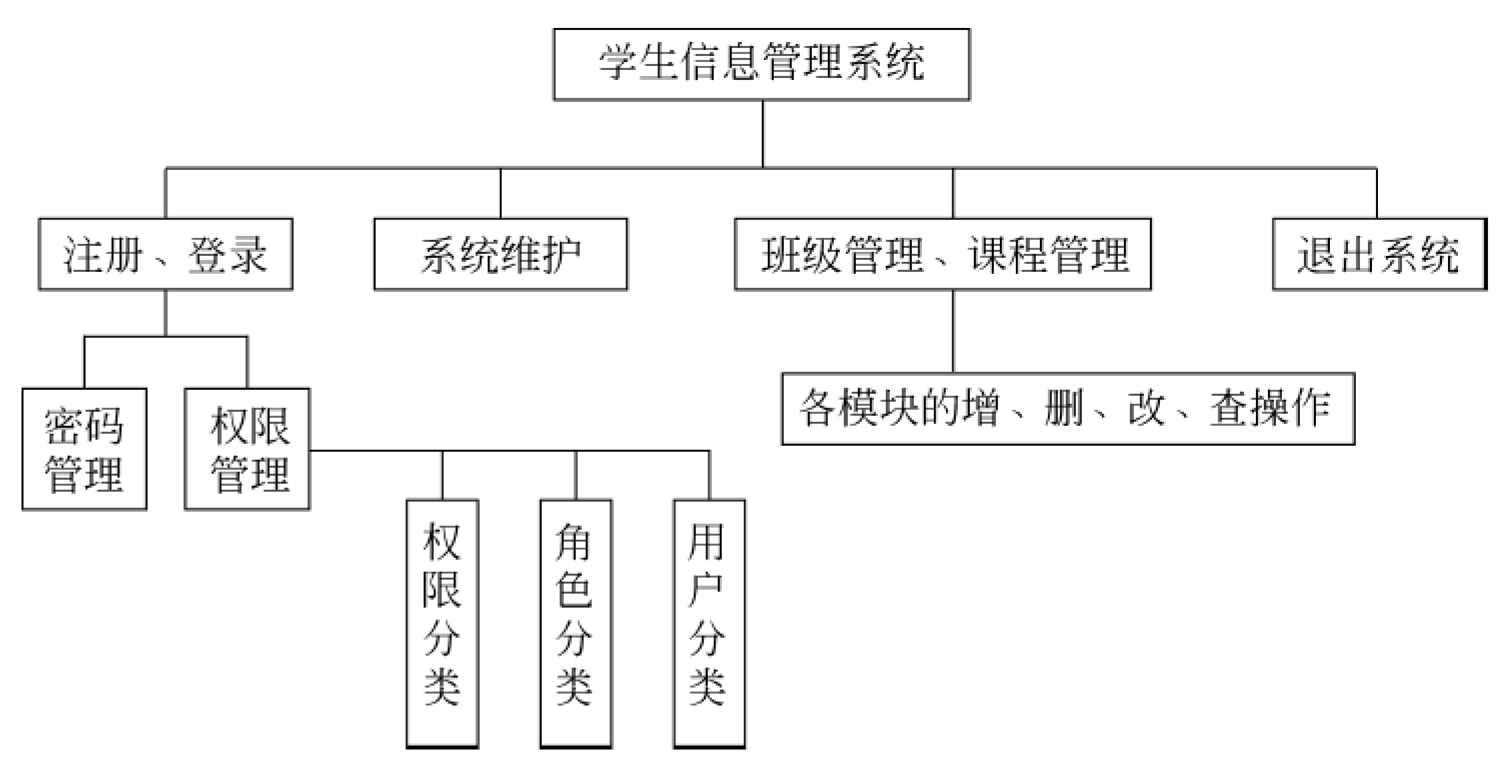


图 7-1　学生信息管理系统结构框架

7.2　数据库设计：系统使用 mysql 数据库

根据系统功能设计的要求以及功能模块的划分，对于系统用户信息数据库，可以列出以下数据项和数据结构。

(1) 学生信息表(见表 7-1)

表名称标识：Student 表。

数据来源：学生信息录入模块进行录入。

表 7-1　学生信息表(Student)

字 段 名	字 段 类 型	长度	主/外键	字段值约束	对应中文名
Student_id	int	4	P	Not null	学号
Student_name	nvarchar	10		Not null	姓名
Sex	char	2			性别
Birth	smalldatetime	4			出生年月
Nation	char	8			民族
Class_id	int	4		Not null	班级号
Entrance_date	smalldatetime			Not null	入学时间
home	nvarchar	40			家庭地址
politic	char	10			政治面貌
ID	nvarchar	18			身份证号
Job	nvarchar	20			职位
specialty	nvarchar	20			所学专业

（2）学生成绩表（见表 7-2）

表名称标识：Student_course 表。

数据来源：学生成绩录入模块进行录入。

表 7-2　学生成绩表（Student_course）

字 段 名	字段类型	长度	主/外键	字段值约束	对应中文名
Course_id	int	4	P	Not null	课程号
Student_id	int	4	P	Not null	学号
Grade	float	8		Not null	成绩
SC_semester	Smallint	2		Not null	学期
School_year	Smallint	2		Not null	学年

（3）班级表（见表 7-3）

表名称标识：class 表。

数据来源：班级管理模块进行录入。

表 7-3　班级表（class）

字 段 名	字段类型	长度	主/外键	字段值约束	对应中文名
Class_id	int	4	P	Not null	班级号
Grade	char	10			年级
Class_name	nvarchar	40		Not null	班级名称
SumStu	int	4			班级人数
MaxNum	int	4			最大人数

（4）课程表（见表 7-4）

表名称标识：course 表。

数据来源：课程管理模块进行录入。

表 7-4　课程表（course）

字 段 名	字段类型	长度	主/外键	字段值约束	对应中文名
Course_id	int	4	P	Not null	课程号
Course_name	Credit	20		Not null	课程名
Credit	Smallint	2			学分

（5）用户表（见表 7-5）

表名称标识：Syuser 表。

数据来源：权限管理模块进行录入。

表 7-5　用户表(Syuser)

字 段 名	字段类型	长度	主/外键	字段值约束	对应中文名
User_id	char	10	P	Not null	用户编号
User_name		10		Not null	用户名
User_role		10			用户角色
Password		8			密码

7.3　项目功能的实现

7.3.1　知识准备

1. Android 网络通信概述

Android 与服务器通信通常采用 HTTP 通信方式和 Socket 通信方式,而 HTTP 通信方式又分为 get 和 post 两种方式。本项目中使用的是 HTTP 通信方式。

(1) HTTP 协议

HTTP (HyperText Transfer Protocol)是 Web 联网的基础,也是手机联网常用的协议之一,HTTP 协议是建立在 TCP 协议上的一种协议。

HTTP 连接最显著的特点是客户端发送的每次请求都需要服务器回送响应,在请求结束后,会主动释放连接。从建立连接到关闭连接的过程称为"一次连接"。在 HTTP 1.0 中,客户端的每次请求都要求建立一次单独的连接,在处理完本次请求后,就自动释放连接。HTTP 1.1 则可以在一次连接中处理多个请求,并且多个请求可以重叠进行,不需要等待一个请求结束后再发送下一个请求。

由于 HTTP 在每次请求结束后都会主动释放连接,因此 HTTP 连接是一种"短连接",要保持客户端程序的在线状态,需要不断地向服务器发起连接请求。通常的做法是即使不需要获得任何数据,客户端也保持每隔一段固定的时间向服务器发送一次"保持连接"的请求,服务器在收到该请求后对客户端进行回复,表明知道客户端"在线"。若服务器长时间无法收到客户端的请求,则认为客户端"下线",若客户端长时间无法收到服务器的回复,则认为网络已经断开。

基于 HTTP 1.0 协议的客户端在每次向服务器发出请求后,服务器就会向客户端返回响应消息,在确认客户端已经收到响应消息后,服务端就会关闭网络连接。在这个数据传输过程中,并不保存任何历史信息和状态信息,因此,HTTP 协议也被认为是无状态的协议。

HTTP 1.1 和 HTTP 1.0 相比,最大的区别就是增加了持久连接支持。当客户端使用 HTTP 1.1 协议连接到服务器后,服务器就将关闭客户端连接的主动权交还给客户端,也就是说,只要不调用 Socket 类的 close 方法关闭网络连接,就可以继续向服务器发送 HTTP 请求。

HTTP 连接使用的是“请求——响应”的方式(2 次握手),不仅在请求时需要先建立连接,而且需要客户端向服务器发出请求后,服务器端才能回复数据。

(2) get 方式

get 方式使用的是在 URL 地址中通过“?”间隔,然后以 name=value 的形式给客户端传递参数。

(3) post 方式

post 方式不在 URL 中传递,也正好解决了 get 传输量小、容易篡改及不安全等一系列不足。主要是通过对 HttpURLConnection 的设置,让其支持 post 传输方式,然后通过相关属性传递参数(若需要传递中文字符,则可以通过 URLEncoder 编码,而在获取端采用 URLDecoder 解码即可)。

(4) get 与 post 请求

post 请求可以向服务器传送数据,而且数据被放在 HTML HEADER 内一起传送到服务端 URL 地址,数据对用户不可见。而 get 是把参数数据队列加入提交的 URL 中,值和表单内各个字段一一对应,例如,http://www.baidu.com/s?w=%C4&inputT=2710。

get 传送的数据量较小,不能大于 2KB。post 传送的数据量较大,一般默认为不受限制。但理论上,IIS 4 中最大量为 80KB,IIS 5 中为 100KB。

get 安全性非常低,post 安全性较高。

2. 获取图片

获取图片界面如图 7-2 和图 7-3 所示。



图 7-2 获取图片 1



图 7-3 获取图片 2

关键代码如下：

```
//ip 地址不能用 localhost 和 127.0.0.1
    String path= "https://www.baidu.com/img/bd_logo1.png";
    //1.新建 url 对象
    try {
        URL url= new URL(path);
        //2.与服务器连接
        HttpURLConnection connection= (HttpURLConnection) url.openConnection();
        //3.设置请求方式和超时时间
        connection.setRequestMethod("GET");
        connection.setReadTimeout(5000);
        //4.获取服务器返回的数据流
        InputStream is= connection.getInputStream();
        //5.把流转换成字节,放入图片控件
        byte[] data= Convert.streamToBytes(is);
        Bitmap bm= BitmapFactory.decodeByteArray(data, 0, data.length);
        ImageView iv= (ImageView) findViewById(R.id.imageView1);
        iv.setImageBitmap(bm);
    } catch (Exception e) {
        e.printStackTrace();
    }
```

输入流转换成字节数组。

```
public static byte[] streamToBytes(InputStream is){
        //字节缓冲流
```

```
        ByteArrayOutputStream baos= new ByteArrayOutputStream();
        try {
            byte[] buffer= new byte[1024];
            int len= 0;
            while((len= is.read(buffer)) != - 1){
                baos.write(buffer, 0, len);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
        return baos.toByteArray();
    }
```

3. 获取网页内容

获取网页内容界面如图 7-4 所示。



图 7-4　获取网页内容

关键代码如下：

```
TextView tv= (TextView) findViewById(R.id.tv);
        tv.setText(new String(data));
```

4. 获取网络 xml 数据

获取网络 xml 数据如图 7-5 所示。



图 7-5　获取网络 xml 数据

关键代码如下：

```
//1.获取 list
        final String path="http://192.168.1.125:8080/webpro/getVideos";
        lv=(ListView) findViewById(R.id.listView1);
        new Thread(new Runnable() {

            @Override
            public void run() {
                GetXmlService service=new GetXmlService();
                videos=service.getXmlData(path);
                handler.sendEmptyMessage(0);
            }
        }).start();
//2.服务器端生成 xml 数据
//加入影片
        List<Video> videos=new ArrayList<Video>();
        videos.add(new Video(110, "速度与激情 7", 120));
        videos.add(new Video(120, "万物生长", 100));
        videos.add(new Video(150, "战狼", 130));

        request.setAttribute("data", videos);
        request.getRequestDispatcher("show.jsp").forward(request, response);
<?xml version="1.0" encoding="UTF-8"?>
<%@page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<videos>
    <c:forEach items="${data}" var="v">
    <video id="${v.id}">
```

```
            <title>${v.title}</title>
            <timelength>${v.timelength}</timelength>
        </video>
        </c:forEach>
</videos>
```

```
//3.使用 pull 解析 xml
public List<Video> getXmlData(String path){
        List<Video> videos=null;
        Video video=null;

        try {
            URL url=new URL(path);
            HttpURLConnection connection=(HttpURLConnection)
            url.openConnection();
            connection.setReadTimeout(5000);
            connection.setRequestMethod("GET");
            InputStream is=connection.getInputStream();
            //解析 xml
            XmlPullParser parser=Xml.newPullParser();
            parser.setInput(is, "UTF-8");
            int event=parser.getEventType();
            while(event!=XmlPullParser.END_DOCUMENT){
                switch (event) {
                case XmlPullParser.START_DOCUMENT:
                    videos=new ArrayList<Video>();
                    break;
                case XmlPullParser.START_TAG:
                    String name=parser.getName();
                    if(name.equals("video")){
                        video=new Video();
                        video.setId(Integer.parseInt(parser.getAttribute-Value(0)));
                    }else if(name.equals("title")){
                        video.setTitle(parser.nextText().toString());
                    }else if(name.equals("timelength")){
                        video.setTimelength(Integer.parseInt(parser.nextText().toString()));
                    }
                    break;
                case XmlPullParser.END_TAG:
                    String tag=parser.getName();
                    if(tag.equals("video")){
                        videos.add(video);
                    }
                    break;
                case XmlPullParser.END_DOCUMENT:
                    return videos;
                default:
                    break;
                }
```

```
                event= parser.next();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        return videos;
}
```

5. 获取网络 json 数据

获取网络 json 数据如图 7-6 所示。



图 7-6 获取网络 json 数据

关键代码如下：

```
//1.服务端生成 jsonstring
String data= JSONArray.toJSONString(videos);

//2.解析 jsonString
        byte[] data= Convert.streamToBytes(is);
        String jsonString= new String(data);
        System.out.println(jsonString);
        videos= JSON.parseArray(jsonString, Video.class);
```

6. 向 Web 端发送数据

向 Web 端发送数据如图 7-7 和图 7-8 所示。

关键代码如下：



图 7-7　向 Web 端发送数据 1



图 7-8　向 Web 端发送数据 2

```
//1.Android端 get方式发送数据
//   http://192.168.4.200/webpro/sendGet?name=tom&pwd=123
    public void sendGet(StringBuffer path, String user,String pwd){
        try {
```

```
                Map<String, String> map=new HashMap<String, String>();
                map.put("name", user);
                map.put("pwd", pwd);
                //遍历 map 的 key
                for (Map.Entry<String, String> entry :map.entrySet()) {
                    path.append(entry.getKey()).append("=").
                    append(URLEncoder.encode(entry.getValue(), "utf-8")).
                    append("&");
                }
                path.deleteCharAt(path.length()-1);
                URL url=new URL(path.toString());
                HttpURLConnection connection=(HttpURLConnection)
                url.openConnection();
                connection.setReadTimeout(5000);
                connection.setRequestMethod("GET");
                //返回状态码
                int code=connection.getResponseCode();
                System.out.println(code);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
//2.服务器端接收 get 方式发送的数据
public void doGet(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        String name=request.getParameter("name");
        String pwd=request.getParameter("pwd");
        System.out.println(name+"-->"+pwd);
}
//3.android端 post 方式发送数据
public void sendpost(String path,Map<String, String> data){
        try {
            //name=jack&pwd=123456
            //遍历 map 的 key
            StringBuffer sb=new StringBuffer();
            for (Map.Entry<String, String> entry :data.entrySet()) {
                sb.append(entry.getKey()).append("=").
                append(URLEncoder.encode(entry.getValue(),
                "utf-8")).append("&");
            }
            sb.deleteCharAt(sb.length()-1);
            System.out.println(sb);
            byte[] bs=sb.toString().getBytes();
            //使用 post 方式发送
            URL url=new URL(path);
            HttpURLConnection connection=(HttpURLConnection)
            url.openConnection();
            connection.setRequestMethod("POST");
```

```
                connection.setReadTimeout(5000);
                //设置输出
                connection.setDoOutput(true);
                //设置头信息
                connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
                connection.setRequestProperty("Content-Length",String.valueOf(bs.length));
                //获取 connection 的输出流
                OutputStream os=connection.getOutputStream();
                os.write(bs);
                os.flush();
                os.close();
                //使用 post 方式发送数据需要接收返回值,否则数据不能发送
                int code=connection.getResponseCode();
                System.out.println(code);
            } catch (Exception e) {
                e.printStackTrace();
            }
    }
    //4.服务器端接收 post 数据
    public void doPost(HttpServletRequest request, HttpServletResponse response)
                throws ServletException, IOException {
            response.setContentType("text/html");
            request.setCharacterEncoding("UTF-8");
            String name=request.getParameter("name");
            String pwd=request.getParameter("pwd");
            System.out.println(name+"-->"+pwd);
    }

    //5.使用 Apache 接口发送数据
```

对于大部分应用程序而言 JDK 本身提供的网络功能已远远不够,这时就需要 Android 提供的 Apache HttpClient。它是一个开源项目,功能更加完善,为客户端的 Http 编程提供高效、最新、功能丰富的工具包支持

```
    //name=jack&pwd=123456
        public void sendHttpClient(String path,Map<String, String> data){
            try {
                List<NameValuePair> pairs=new ArrayList<NameValuePair>();
                //遍历 map
                for (Map.Entry<String, String> entry : data.entrySet()) {
                    pairs.add(new BasicNameValuePair(entry.getKey(),
                    entry.getValue()));
                }
                //转换字符编码
                UrlEncodedFormEntity data1=new UrlEncodedFormEntity(pairs,
                "utf-8");
                //发送数据
                HttpPost post=new HttpPost(path);
                post.setEntity(data1);
                //模拟浏览器
                HttpClient client=new DefaultHttpClient();
```

```
            HttpResponse response= client.execute(post);
            //得到返回的状态码
            System.out.println(response.getStatusLine().getStatusCode());

        } catch (Exception e) {
            e.printStackTrace();
        }

}

//6.Android发送 xml 数据
<?xml version="1.0" encoding="UTF-8"?>
<persons>
    <person id="23">
        <name>李磊</name>
        <age>30</age>
    </person>
    <person id="20">
        <name>韩梅梅</name>
        <age>25</age>
    </person>
</persons>
public void sendXml(View view){
        //1.加载 xml
        new Thread(new Runnable() {
            @Override
            public void run() {
                try{
                    InputStream is= this.getClass().getClassLoader().
                    getResourceAsStream("persons.xml");
                    byte[] data= Convert.streamToBytes(is);
                    //2.使用 http协议发送 post
                    String path= "http://192.168.0.114:8080/webpro/sendxml";
                    URL url= new URL(path);
                    HttpURLConnection conn= (HttpURLConnection)
                    url.openConnection();
                    conn.setRequestMethod("POST");
                    conn.setReadTimeout(5000);
                    conn.setDoOutput(true);
                    //设置文件类型
                    conn.setRequestProperty("Content-Type", "text/xml;
                    charset=utf-8");
                    conn.setRequestProperty("Content-Length", String.valueOf
                    (data.length));
                    OutputStream os= conn.getOutputStream();
                    os.write(data);
                    os.flush();
                    os.close();
                    int code= conn.getResponseCode();
```

```
                    System.out.println(code);
                }catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }).start();
}

//7.服务器端接收 xml 数据
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");
        request.setCharacterEncoding("UTF-8");
        InputStream is= request.getInputStream();
        byte[] data= Convert.streamToBytes(is);
        String xml= new String(data,"UTF-8");
        System.out.println(xml);
}
```

7. Android Butterknife 框架

Android Butterknife 是 Android 上通过依赖注入(DI)获取 xml 界面控件、设置事件的一个框架。使用步骤如下:

(1) 准备阶段,先到官网(http://jakewharton.github.io/butterknife/)下载 jar 包。

(2) 把下载下来的 jar 包,放到项目的 libs 下,就会自动导入项目。

(3) 配置 eclips。

(4) 注解。

XML 部分代码如下:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

   <TextView
        android:id="@+id/tv_test"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/hello_world"
        tools:context=".MainActivity" />

</RelativeLayout>
```

Java 部分代码如下:

```
package com.msquirrel.main;

import butterknife.ButterKnife;
import butterknife.InjectView;
import butterknife.OnClick;
import android.os.Bundle;
import android.app.Activity;
import android.widget.TextView;

public class MainActivity extends Activity {

    @InjectView(R.id.tv_test)
    TextView tvTest;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ButterKnife.inject(this);
        tvTest.setText("test");
    }

    @OnClick(R.id.tv_test)
    public void sayHi() {
        tvTest.setText("Hello!");
    }
}
```

7.3.2 Splash 界面设计

Splash 界面设计如图 7-9 所示。

每个 Android 应用启动之后都会出现一个 Splash 启动界面，显示产品的 LOGO、公司的 LOGO 或者开发者信息。如果应用程序启动时间比较长，那么启动界面就非常重要，可以让用户耐心等待一段时间。

通过制作 Splash 界面可实现如下功能。

(1) 突出产品 LOGO、产品名称、产品主要特色。

(2) 注明产品的版本信息。

(3) 注明公司信息或者开发者信息。

需要注意，大多数的 Splash 界面都会在一定时间后切换到下一个界面，在这段时间里，可以对系统状况进行检测，比如网络是否通畅、电源是否充足，或者预先加载相关数据。为了能让启动界面展现时间固定，需要计算执行以上预处理任务所花费的时间，启动界面 SLEEP 的时间＝固定时间－预处理任务时间。

XML 代码如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```



图 7-9 Splash 界面设计

```
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=".SplashActivity"
android:background="@drawable/splash_bg"
android:id="@+id/rl">

<TextView
    android:id="@+id/tv_version"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="162dp"
    android:text="@string/hello_world"
    android:textSize="20sp" />

<ImageView
    android:id="@+id/imageView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/tv_version"
    android:layout_alignLeft="@+id/tv_version"
    android:layout_marginBottom="28dp"
```

```
        android:src="@drawable/cat" />

</RelativeLayout>
```

7.3.3　系统升级

系统升级如图 7-10 和图 7-11 所示。



图 7-10　系统升级 1



图 7-11　系统升级 2

具体步骤如下：

（1）检测当前版本的信息 AndroidManifest.xml→manifest→android:versionName。

（2）从服务器获取版本号(版本号存在于 xml 文件中)，并与当前检测到的版本进行匹配，如果不匹配，提示用户进行升级，如果匹配则进入程序主界面。

（3）当提示用户进行版本升级时，如果用户单击“确定”按钮，系统将自动从服务器上下载并进行自动升级，如果单击“取消”按钮将进入程序主界面。

升级流程如图 7-12 所示。

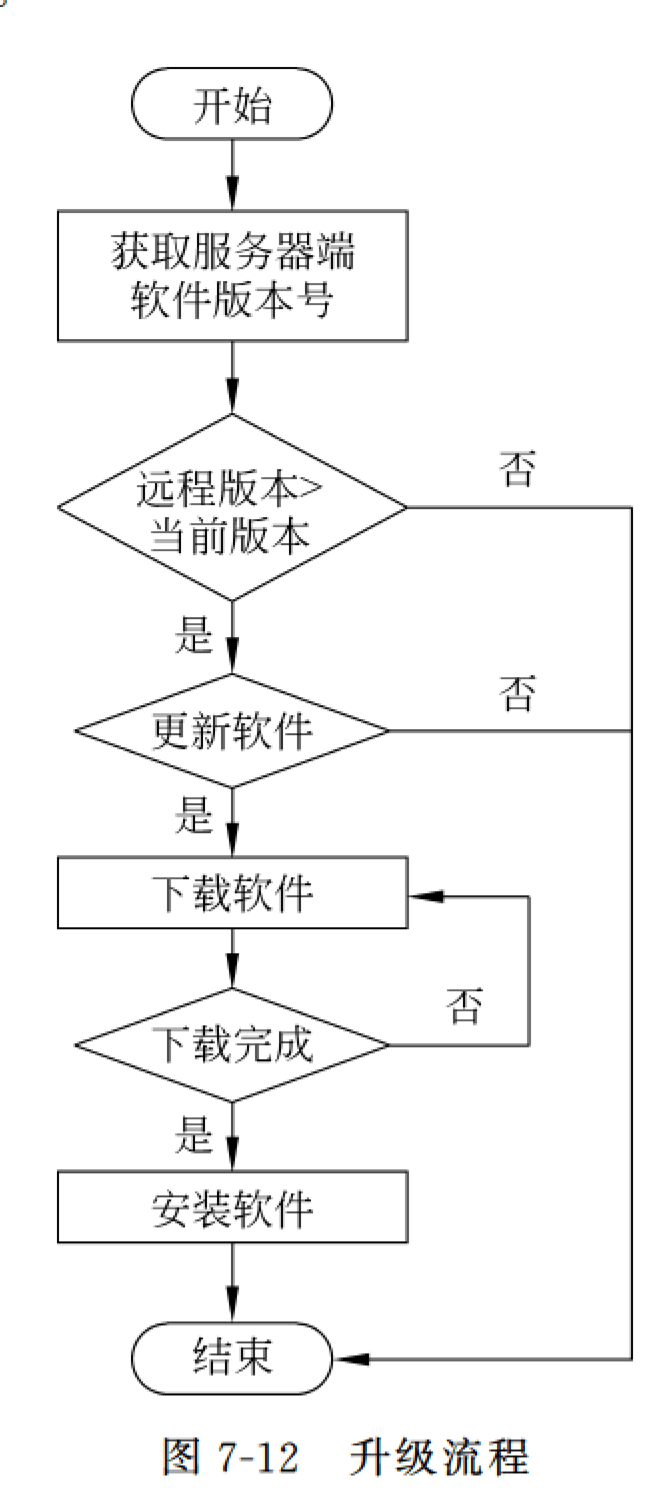


图 7-12 升级流程

关键代码如下：

```
//弹出对话框
    public void showAlertDiolog(){
        AlertDialog.Builder builder=new Builder(this);
        builder.setTitle("升级提醒");
        builder.setMessage(info.getDes());
        builder.setIcon(R.drawable.logo);
        builder.setNegativeButton("确定", new OnClickListener() {

            @Override
            public void onClick(DialogInterface dialog, int which) {
                pd.show();
                new Thread(new Runnable() {

                    @Override
```

```
                        public void run() {
                            //下载文件
                            DownLoadFile downLoadFile= new DownLoadFile();
                            file= downLoadFile.doloadFile(info.getApkurl(),
                            "/mnt/sdcard/new.apk", pd);
                            handler.sendEmptyMessage(1);

                        }
                    }).start();

                }
            });
            builder.setPositiveButton("取消", new OnClickListener() {

                @Override
                public void onClick(DialogInterface dialog, int which) {
                    loadLogin();
                }

            });
            //显示对话框
            builder.create().show();
        }

//下载服务器 apk 文件
public class DownLoadFile {
    public File doloadFile(String downloadPath,String savePath,
    ProgressDialog pd) {
        File outFile= new File(savePath);
        try {
            URL url= new URL(downloadPath);
            HttpURLConnection conn= (HttpURLConnection)url.openConnection();
            conn.setReadTimeout(5000);
            conn.setRequestMethod("GET");
            if(conn.getResponseCode()==200) {
                //下载
                InputStream is= conn.getInputStream();
                //设置下载进度条
                int  total= conn.getContentLength();
                pd.setMax(total);
                FileOutputStream fos= new FileOutputStream(outFile);
                byte[] buffer= new byte[1024];
                int len= 0;
                int progress= 0;
                while((len= is.read(buffer))!=-1) {
                    fos.write(buffer, 0, len);
                    progress+=len;
                    pd.setProgress(progress);
                    //暂停一下
```

```
                    Thread.sleep(50);
                }
                fos.flush();
                fos.close();
                return outFile;
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

7.3.4 安装升级文件

安装升级文件如图 7-13 和图 7-14 所示。

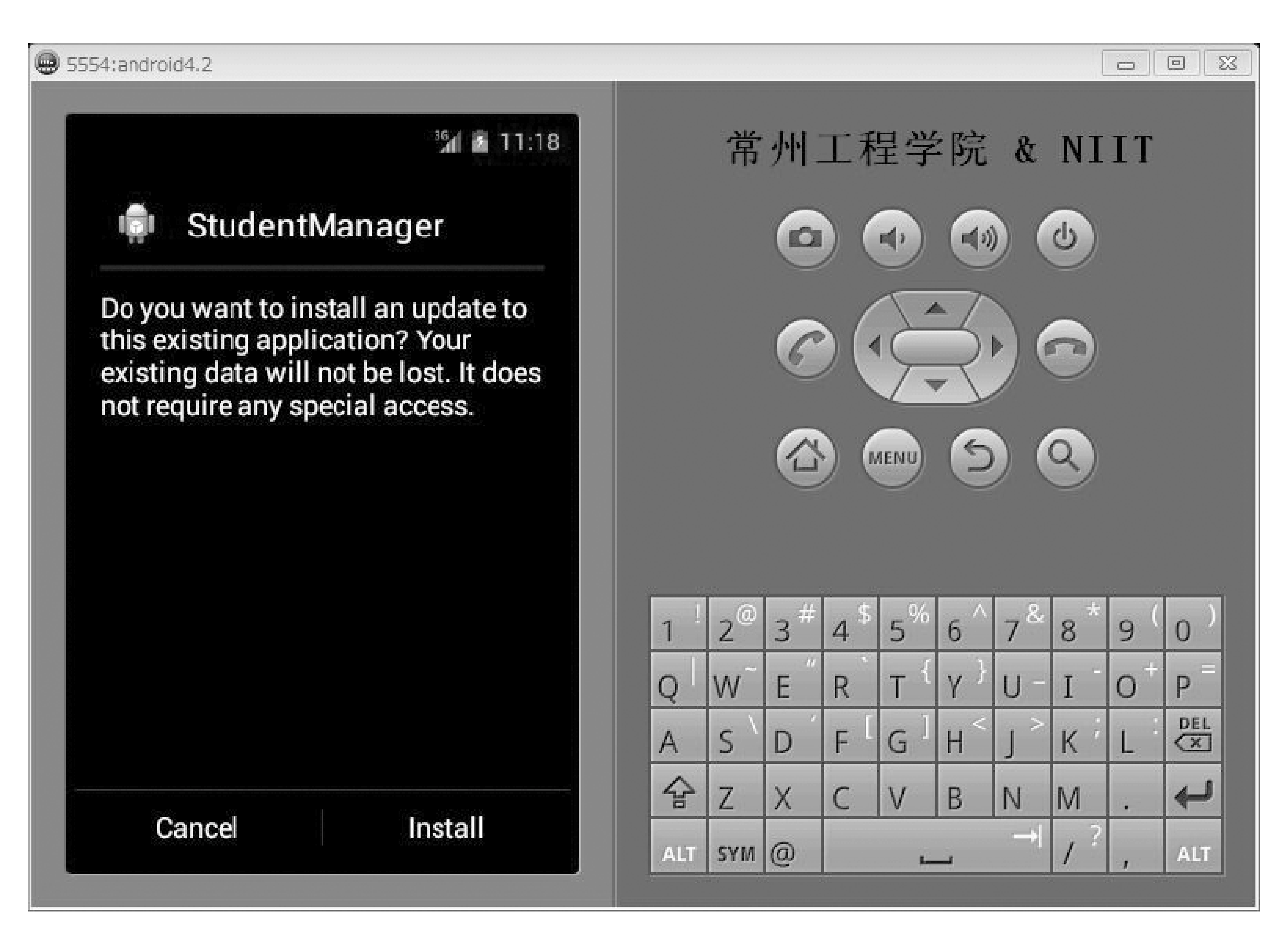


图 7-13 安装升级文件 1

关键代码如下：

```
//程序中安装文件
    public void install(File file){
        Intent intent= new Intent();
        intent.setAction(Intent.ACTION_VIEW);
        intent.setDataAndType(Uri.fromFile(file), "application/vnd.android.
        package- archive");
        startActivity(intent);
    }
```



图 7-14 安装升级文件 2

7.3.5 注册、登录功能

注册、登录界面如图 7-15 和图 7-16 所示。

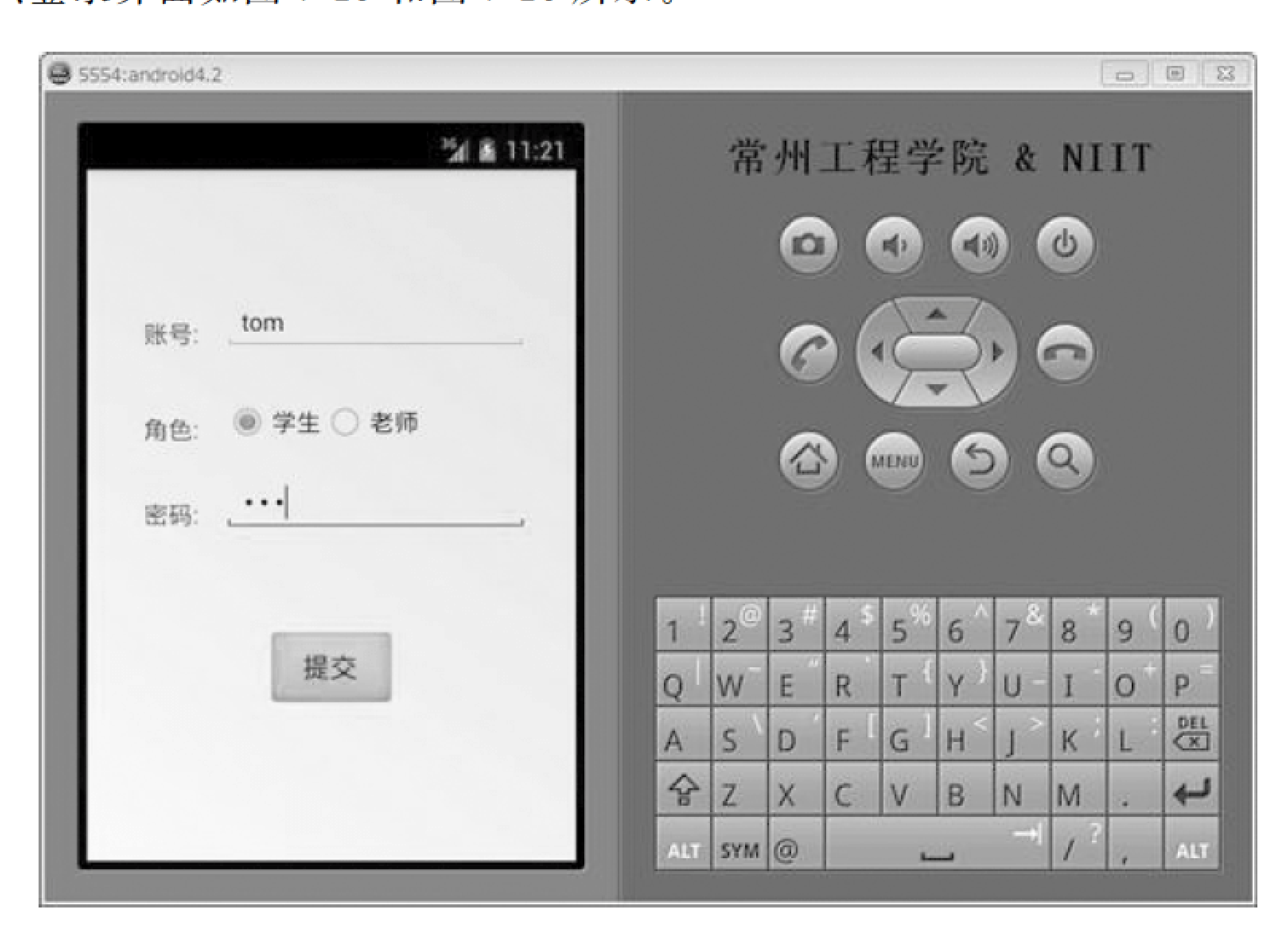


图 7-15 注册界面

实现的效果说明如下。

(1) 注册效果：用户如没有账号，则单击登录界面的“没有账号？注册”。

(2) 登录效果：应用程序判断当前用户还未登录，用户输入账号和密码信息后，传到



图 7-16 登录界面

服务器验证，验证成功后，Toast 弹出成功信息，并转到其他界面。

进入注册界面，用户输入注册信息，单击“提交”按钮，注册成功后，弹出 Toast 信息“注册成功”，完成注册后，转到登录界面。

功能实现说明：

(1) 注册界面：输入注册信息，实现注册功能。

(2) 登录界面：输入登录信息，实现登录功能，转到注册界面。

1. 注册信息中的密码经过 MD5 加密

MD5 即 Message-Digest Algorithm 5(信息-摘要算法 5)，用于确保信息传输完整一致，是计算机广泛使用的杂凑算法之一(又译摘要算法、哈希算法)，不可逆，主流编程语言普遍已有 MD5 实现，在 Java 中广泛使用 MD5 对重要信息进行加密。Java 中实现代码如下：

```
public static String encode(String pwd) {
        try {
            MessageDigest  digest=MessageDigest.getInstance("MD5");
            byte[]  bytes=digest.digest(pwd.getBytes());
            StringBuffer sb=new  StringBuffer();
            for(int i=0;i<bytes.length;i++){
                String s=Integer.toHexString(0xff&bytes[i]);

                if(s.length()==1){
                    sb.append("0"+s);
                }else{
                    sb.append(s);
                }
```

```
            }

            return sb.toString();
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
            throw new RuntimeException("buhuifasheng");
        }
}
```

2. SharedPreferences

(1) SharedPreferences 简介

为了保存软件的设置参数，Android 平台提供了一个 SharedPreferences 类，它是一个轻量级的存储类，特别适用于保存软件配置参数。使用 SharedPreferences 保存数据，其背后是用 xml 文件存放数据，文件存放在/data/data/<package name>/shared_prefs 目录下。

(2) 获取 SharedPreferences 对象方法

① SharedPreferences pre = Context. getSharedPreferences(String name, int mode);

注：name 为本组件的配置文件名(如果想要与本应用程序的其他组件共享此配置文件，可以用这个名字检索配置文件，在这里要特别注意，因为在 Android 中已经确定了 SharedPreferences 是以 xml 形式保存，所以，在填写文件名参数时，不要给定. xml 后缀，只要直接写上文件名。它会直接被保存在/data/data/<package name>/shared_prefs 路径下，它是采用键值对的形式保存参数。当需要获得某个参数值时，按照参数的键索引)。

② SharedPreferences pre = Activity. getPreferences(int mode);

注：配置文件仅可以被调用的 Activity 使用。mode 为操作模式，默认的模式为 0 或 MODE_PRIVATE，还可以使用 MODE_APPEND、MODE_WORLD_READABLE 和 MODE_WORLD_WRITEABLE。

③ SharedPreferences pre = PreferenceManager. getDefaultSharedPreferences(Context);

注：每个应用都有一个默认的配置文件 preferences. xml，使用 getDefaultShared-Preferences 获取。

(3) SharedPreferences 使用步骤

SharedPreferences 使用非常简单，能够轻松地存放数据和读取数据。SharedPreferences 只能保存简单类型的数据，如 String、int 等。一般会将复杂类型的数据转换成 Base64 编码，然后将转换后的数据以字符串的形式保存在 xml 文件中，再用 SharedPreferences 保存。

使用 SharedPreferences 保存 key-value 对的步骤如下：

① 获得 SharedPreferences 对象。

② 获得 SharedPreferences. Editor 对象。

③ 通过 SharedPreferences. Editor 接口的 putXxx()方法存放 key-value 对(其中 Xxx 表示不同的数据类型，如字符串类型的 value 需要用 putString()方法)。

④ 通过 SharedPreferences. Editor 接口的 commit()方法保存 key-value 对(commit 方法相当于数据库事务中的提交(commit)操作)。

(4) 存储数据和读取数据的流程

存储数据信息的流程如下:

① 打开名为 configuration 的配置文件,如果存在则打开它,否则创建新的名为 configuration 的配置文件。

```
SharedPreferences sharedPreferences= getSharedPreferences("configuration", 0);
```

② 让 sharedPreferences 处于编辑状态。

```
SharedPreferences.Editor editor= sharedPreferences.edit();
```

③ 存放数据。

```
editor.putString("name", "harvey");
```

④ 完成提交。

```
editor.commit();
```

读取数据信息的流程如下:

① 打开名为 configuration 的配置文件。

```
SharedPreferences sharedPreferences= getSharedPreferences("configuration", 0);
```

② 获取数据。

```
String name= sharedPreferences.getString("name","默认值");
```

以上就是 Android 中 SharedPreferences 的使用方法,其中创建的配置文件存放位置可以在 Eclipse 中查看:

```
DDMS---File Explorer---data/data/<package name>/shared_prefs/
configuration.xml
```

关键代码如下:

```
//第一次登录成功后需要把信息写入 sharedpreference
            SharedPreferences preferences= getSharedPreferences("account",
            Context.MODE_PRIVATE);
            //获取编辑器
            Editor editor= preferences.edit();
            editor.putString("user", user);
            editor.putString("pwd", pwd);
            editor.putString("role", msg2.getMsg());
            //提交
            editor.commit();
//通过 SharedPreferences 判断是否要登录
public boolean isLogin(){
```

```
        SharedPreferences preferences=getSharedPreferences("account", Context.MODE_PRIVATE);
        String user=preferences.getString("user", "");
        String pwd=preferences.getString("pwd", "");
        if(!user.equals("") && !pwd.equals("")){
            return false;           //不需登录
        }
        return true;
    }
```

7.3.6　学生信息管理功能

1. Fragment

Android 是在 Android 3.0(API level 11)开始引入 Fragment 的。

可以把 Fragment 想象成 Activity 中的模块,这个模块有自己的布局,有自己的生命周期,单独处理自己的输入,在 Activity 运行时可以加载或者移除 Fragment 模块。

可以把 Fragment 设计成可以在多个 Activity 中复用的模块。

当开发的应用程序同时适用于平板电脑和手机时,可以利用 Fragment 实现灵活的布局,改善用户体验,如图 7-17 所示。

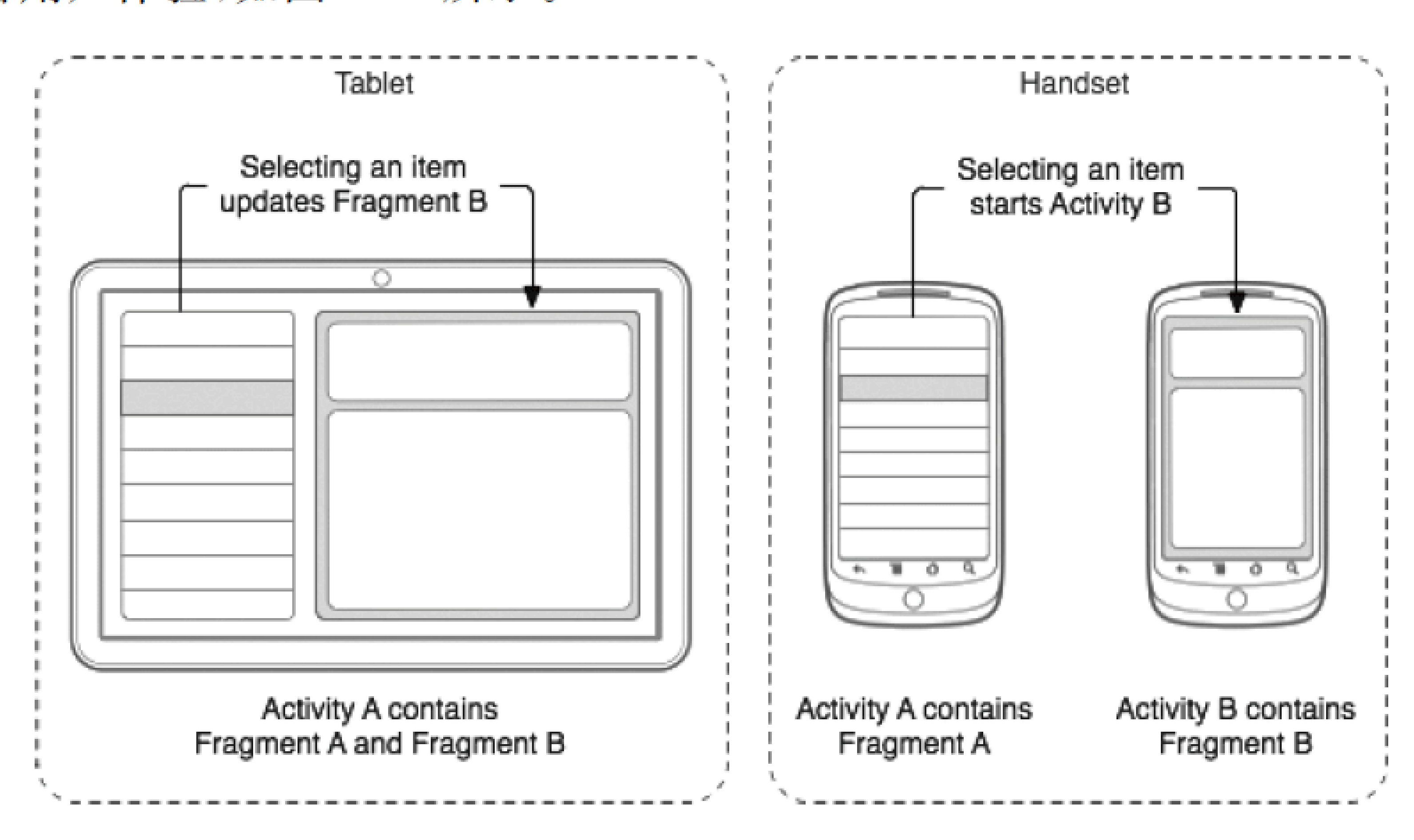


图 7-17　Fragment

2. Fragment 的生命周期

因为 Fragment 必须嵌入在 Acitivity 中使用,所以 Fragment 的生命周期和它所在的 Activity 是密切相关的。

如果 Activity 是暂停状态,其中所有的 Fragment 都是暂停状态;如果 Activity 是停止状态,这个 Activity 中所有的 Fragment 都不能被启动;如果 Activity 被销毁,那么它其中的所有 Fragment 都会被销毁。

但是，当 Activity 在活动状态时，可以独立控制 Fragment 的状态，比如添加或者移除 Fragment。

当这样进行 Fragment Transaction（转换）时，可以把 Fragment 放入 Activity 的 back stack 中，这样用户就可以进行返回操作。Fragment 的生命周期如图 7-18 所示。

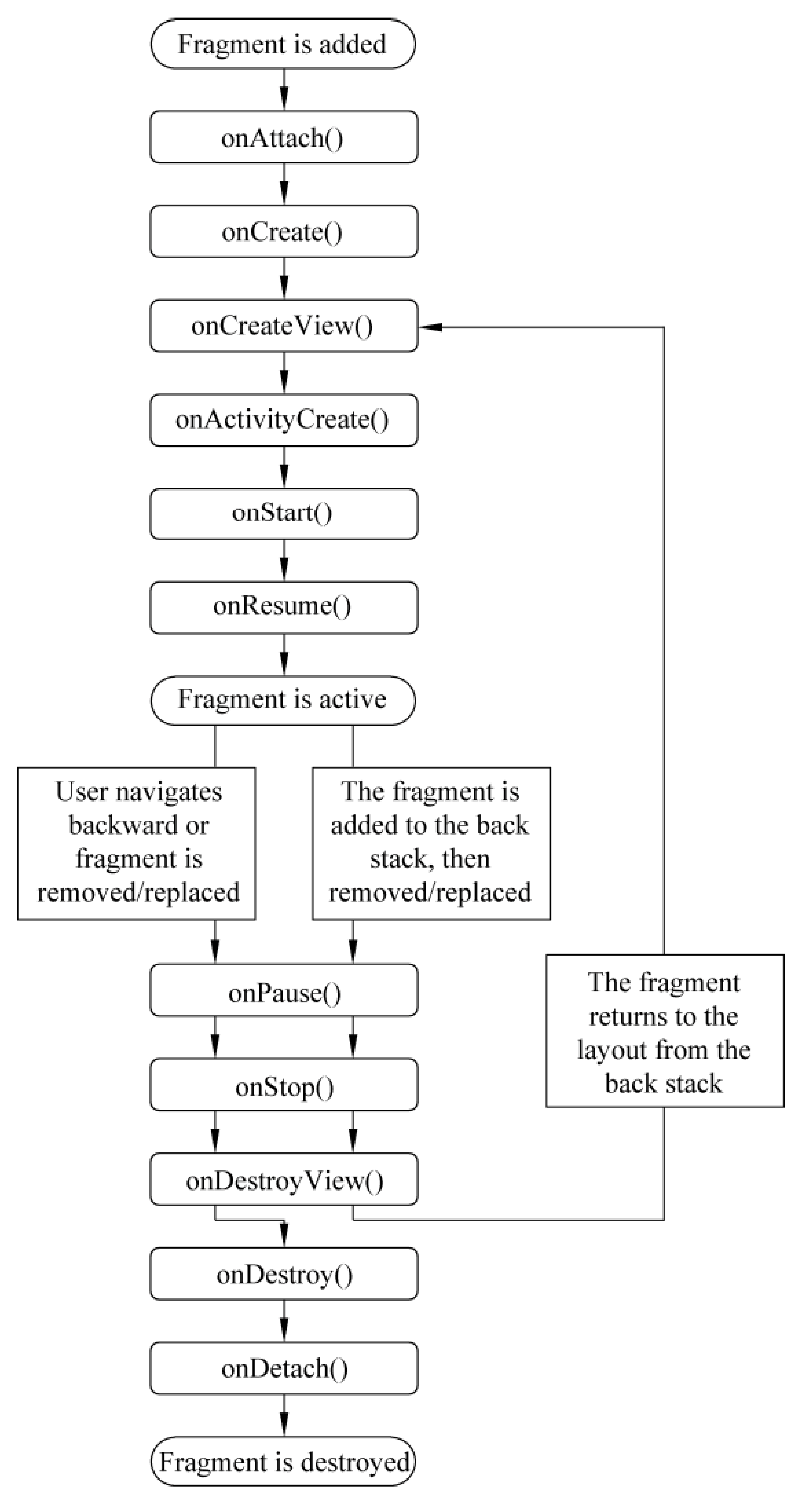


图 7-18 Fragment 的生命周期

Fragment 的控制，主要是切换 View 和页面替换等操作，如何获取 Fragment 的管理对象，以及与 Activity 的通信方式。

（1）管理 Fragment

要在 Activity 中管理 Fragment，需要使用 FragmentManager，通过调用 Activity 的 getFragmentManager()取得实例。

① 使用 findFragmentById()(用于在 Activity layout 中提供一个 UI 的 Fragment)或 findFragmentByTag()(适用于有或没有 UI 的 Fragment)获取 Activity 中存在的 Fragment。

② 使用 popBackStack()(模拟用户按下 BACK 按钮),将 Fragment 从后台堆栈中弹出。

③ 使用 addOnBackStackChangeListener()注册一个监听后台堆栈变化的 listener。

(2) 处理 Fragment 事务

关于在 Activity 中使用 Fragment 的一个很强的特性是:根据用户的交互情况,对 Fragment 进行添加、移除、替换以及执行其他动作。提交给 Activity 的每一套变化被称为一个事务,可以使用在 FragmentTransaction 中的 API 处理。也可以保存每一个事务到一个 Activity 管理的 backstack,允许用户经由 Fragment 的变化往回导航(类似于通过 Activity 往后导航)。

从 FragmentManager 获得一个 FragmentTransaction 实例的代码如下:

```
FragmentManager fragmentManager=getFragmentManager();
FragmentTransaction fragmentTransaction=fragmentManager.beginTransaction();
```

每一个事务都是同时要执行的一套变化。可以在一个给定的事务中设置用户执行的所有变化,使用诸如 add()、remove()和 replace()。然后,要给 Activity 应用事务,必须调用 commit()。

在调用 commit()之前,用户可能调用 addToBackStack(),将事务添加到一个 Fragment 事务的 backstack。这个 back stack 由 Activity 管理,并允许用户通过按下 BACK 按钮返回到前一个 Fragment 状态。

代码如下:

```
//创建修改实例
Fragment newFragment=newExampleFragment();
FragmentTransaction transaction=getFragmentManager().beginTransaction();
//Replace whatever is in the fragment_container view with this fragment,
//and add the transaction to the backstack
transaction.replace(R.id.fragment_container,newFragment);
transaction.addToBackStack(null);
//提交修改
transaction.commit();
```

上面是将一个 Fragment 替换为另一个,并在后台堆栈中保留之前的状态。在这个例子中,newFragment 替换当前 layout 容器中的由 R. id. fragment_container 标识的 fragment。通过调用 addToBackStack(),replace 事务被保存到 back stack,因此用户可以回退事务,并通过按下 BACK 按钮带回前一个 fragment。

如果添加多个变化到事务(例如 add()或 remove())并调用 addToBackStack(),然后在调用 commit()之前的所有应用的变化会被作为一个单个事务添加到后台堆栈,

BACK 按钮会将它们一起回退。添加变化到 FragmentTransaction 的顺序不重要，除以下例外。

① 必须最后调用 commit()。

② 如果添加多个 Fragment 到同一个容器，那么添加的顺序决定了它们在 view hierarchy 中显示的顺序。

当执行一个移除 Fragment 的事务时，如果没有调用 addToBackStack()，那么当事务提交后，那个 Fragment 会被销毁，并且用户不能导航返回。有鉴于此，当移除一个 Fragment 时，如果调用了 addToBackStack()，那么 Fragment 会被停止，如果用户导航回来，它将会被恢复。另外，对于每一个 Fragment 事务，可以应用一个事务动画，通过在提交事务之前调用 setTransition()实现。

调用 commit()并不立即执行事务。恰恰相反，它将事务安排排期，一旦准备好，就在 activity 的 UI 线程上运行(主线程)。如果有必要，无论如何，可以从用户的 UI 线程调用 executePendingTransactions()立即执行由 commit()提交的事务。此操作通常不必要，除非事务是其他线程中任务的一个从属。

警告：只能在 Activity 保存它的状态(当用户离开 Activity)之前使用 commit()提交事务。

(3) 与 Activity 通信

尽管 Fragment 被实现为一个独立于 Activity 的对象，并且可以在多个 Activity 中使用，但一个给定的 Fragment 实例是直接绑定到包含它的 Activity 的。特别的 Fragment 可以使用 getActivity()访问 Activity 实例，并且容易地执行比如在 Activity layout 中查找一个 View 的任务。代码如下：

```
View listView= getActivity().findViewById(R.id.list);
```

同样地，Activity 可以通过从 FragmentManager 获得一个到 Fragment 的引用来调用 Fragment 中的方法，使用 findFragmentById()或 findFragmentByTag()。代码如下：

```
ExampleFragment fragment= (ExampleFragment)
getFragmentManager().findFragmentById(R.id.example_fragment);
```

(4) 总结

最后需要说明一下 Fragment 的例子，Android 官方已经提供了 Fragment 使用各种例子，在 SDK 下面的 API Demo 中就包含了 Fragment 的各种例子，需要看的朋友，直接看 API Demo 那个程序就可以了，里面分不同功能，实现了不同的类，可以根据需要查看具体代码。

3. 程序运行

程序运行如图 7-19～图 7-23 所示。



图 7-19　学生班级查询



图 7-20　学生课程查询



图 7-21 学生成绩查询



图 7-22 修改密码



图 7-23　老师管理功能

4. 程序中的主要代码

```
//设置 fragment 页面
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <FrameLayout
        android:id="@+id/realtabcontent"
        android:layout_width="fill_parent"
        android:layout_height="0dp"
        android:background="@color/bigbg"
        android:layout_weight="1" />

    <RadioGroup
        android:id="@+id/tab_rg_menu"
        android:layout_width="fill_parent"
        android:layout_height="63dip"
        android:padding="1dip"
        android:background="@color/white"
        android:orientation="horizontal"
        >
        <RadioButton
            android:id="@+id/tab_rb_1"
```

```
                style="@ style/tab_rb_style"
                android:checked="true"
                android:drawableBottom="@ drawable/tab_selector_weixing" />

            <RadioButton
                android:id="@ + id/tab_rb_2"
                style="@ style/tab_rb_style"
                android:drawableBottom="@ drawable/tab_selector_tongxunlu" />

            <RadioButton
                android:id="@ + id/tab_rb_3"
                style="@ style/tab_rb_style"
                android:drawableBottom="@ drawable/tab_selector_faxian" />
        </RadioGroup>

</LinearLayout>
//使用 fragment
private FragmentTabHost mTabHost;
    private RadioGroup mTabRg;
    private long firstTime=0;
    public List<Fragment> fragments=new ArrayList<Fragment>();
    @ InjectView(R.id.tab_rb_2)RadioButton tab_rb_2;
    @ Override
    protected void onCreate(Bundle savedInstanceState) {

        //TODO Auto-generated method stub
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ButterKnife.inject(this);

        fragments.add(new QueryFragment());
        fragments.add(new ManagerFragment());
        fragments.add(new MyFragment());
        mTabRg=(RadioGroup) findViewById(R.id.tab_rg_menu);

        FragmentTabAdapter tabAdapter=new FragmentTabAdapter(this,
        fragments, R.id.realtabcontent, mTabRg);
        //获取登录用户的角色
        SharedPreferences preferences=this.getSharedPreferences("account",
        Context.MODE_PRIVATE);
            Contans.role=preferences.getString("role", "");
            Contans.s_name=preferences.getString("user", "");

    }

//查询功能
public class QueryFragment extends Fragment{
    String res="";
```

```
Map< String, String> data= new HashMap< String, String> ();
HttpUtils utils= new HttpUtils();
String path= "";
@ InjectView(R.id.lv_data) ListView lv_data;
List< T_Class> out_data= new ArrayList< T_Class> ();
List< T_Course> out_data_course= new ArrayList< T_Course> ();
List< T_Student> out_data_student= new ArrayList< T_Student> ();
List< T_Grade> out_data_grade= new ArrayList< T_Grade> ();
//设置布局文件
@ Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
    View parentView= inflater.inflate(R.layout.query_fragment,
    container, false);
    ButterKnife.inject(this, parentView);
    return parentView;
}
//处理业务
@ Override
public void onViewCreated(View view, Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    Toast.makeText(getActivity(), Contans.role, 1).show();

}

//单击查询班级
@ OnClick(R.id.tv_class)
public void queryClass(View view){
    //根据不同角色查询数据
    if(Contans.role.equals("student")){
        data.put("num", "one");
        data.put("s_name", Contans.s_name);
    }else if(Contans.role.equals("teacher")){
        data.put("num", "all");
    }
    new Thread(new Runnable() {
        @ Override
        public void run() {
            path= getActivity().getResources().getString(R.string.
            getCalss_url);
            res= utils.sendHttpClient(path, data);
            handler.sendEmptyMessage(0);
        }
    }).start();

}

@ OnClick(R.id.tv_course)
public void querycourse(View v)
```

```
{
    data.put("num", "all");
    new Thread(new Runnable() {
        @Override
        public void run() {
            path=getActivity().getResources().getString(R.string.
            getCourse_url);
            res=utils.sendHttpClient(path, data);
            handler.sendEmptyMessage(1);
        }
    }).start();
}

@OnClick(R.id.tv_student)
public void querystudent(View v)
{
    if(Contans.role.equals("student")){
        data.put("num", "one");
        data.put("s_name", Contans.s_name);
    }else if(Contans.role.equals("teacher")){
        data.put("num", "all");
    }
    new Thread(new Runnable() {
        @Override
        public void run() {
            path=getActivity().getResources().getString(R.string.
            getStudent_url);
            res=utils.sendHttpClient(path, data);
            handler.sendEmptyMessage(2);
        }
    }).start();

}

@OnClick(R.id.tv_grade_c)
public void querygrade(View v)
{
    if(Contans.role.equals("student")){
        data.put("num", "one");
        data.put("s_name", Contans.s_name);
    }else if(Contans.role.equals("teacher")){
        data.put("num", "all");
    }
    new Thread(new Runnable() {
        @Override
        public void run() {
            path=getActivity().getResources().getString(R.string.
            getGrade_url);
            res=utils.sendHttpClient(path, data);
```

```
                handler.sendEmptyMessage(3);
            }
        }).start();
    }

    Handler handler=new Handler(){
        public void handleMessage(android.os.Message msg) {
            Gson gson=new Gson();
            try {
                res=new String(res.getBytes("iso-8859-1"), "GBK");
            } catch (UnsupportedEncodingException e) {
                //TODO Auto-generated catch block
                e.printStackTrace();
            }

            switch(msg.what)
            {

            case 0:

                System.out.println(res);
                out_data=gson.fromJson(res, new TypeToken<List<T_Class>>() {
                }.getType());
                ClassApapter apapter=new ClassApapter(out_data, getActivity());
                lv_data.setAdapter(apapter);
                break;
            case 1:
                System.out.println(res);
                out_data_course=gson.fromJson(res, new TypeToken<List<T_
                Course>>() {
                }.getType());
                CourseAdapter adapter=new CourseAdapter(out_data_course,
                getActivity());
                lv_data.setAdapter(adapter);
                break;
            case 2:
                System.out.println(res);
                out_data_student=gson.fromJson(res, new TypeToken<List<T_
                Student>>() {
                }.getType());
                StudentAdapter adapter2=new StudentAdapter(out_data_student,
                getActivity());
                lv_data.setAdapter(adapter2);
                break;
            case 3:
                System.out.println(res);
                out_data_grade=gson.fromJson(res, new TypeToken<List<T_
                Grade>>() {
                }.getType());
```

```
                GradeAdapter adapter3=new GradeAdapter(out_data_grade,
                getActivity());
                lv_data.setAdapter(adapter3);
                break;
            default:
                break;
            }
        }

    };

}
```

7.4 系统运行与效果测试

系统运行与效果测试如图 7-10、图 7-15、图 7-16、图 7-19、图 7-22 和图 7-23 所示。

7.5 本章小结

通过本章项目练习，学习了 Android 的网络通信，MD5 算法、SharedPreferences 和 Fragment 等，了解了项目开发的整体流程。

7.6 项目实践

（1）优化界面。
（2）封装 HTTP 网络通信。
（3）扩充程序功能。

第 8 章

影音播放器的设计及开发

本章的工作目标如下：

(1) 使用 Tomcat 搭建远端服务器并完成图片、分类、名称和影片等资源的请求接口。

(2) 使用 Fragment 搭建可单击 item 切换的界面。

(3) 使用网络请求获取影片的分类、图片、名称和资源等信息。

(4) 根据 url 加载，在线播放影片。

(5) 完成程序运行和效果测试。

8.1 项目分析

本项目的学习和操作主要关注以下几点。

(1) 了解并使用 Tomcat 搭建服务器。

(2) 掌握 MediaPlayer 的基本用法。

(3) 掌握 SurfaceView 的基本用法。

(4) 熟悉网络请求、图片加载和多图片缓存，了解 OOM 出现的原因并能有效地预防 OOM。

(5) 综合以上几点，完成影音播放器的设计及开发。

影音播放器的设计及开发如图 8-1 所示。

1. 首页界面

首页主要为影片分类和列表，列表部分使用了 GridView（网格布局），也可由 RecyclerView 代替 GridView；在底部有一个 ProgressBar 用来显示正在加载。

2. 首页功能

首页功能主要是展示影片信息，首先通过 HttpClient 请求到网络资源，然后通过 Json 解析后使用 Adapter 装载数据，此时需要注意图片的缓存问题。

3. 播放界面

播放界面主要由 VideoView 和 ProgressBar 两个控件组成。外层为 RelativeLayout，

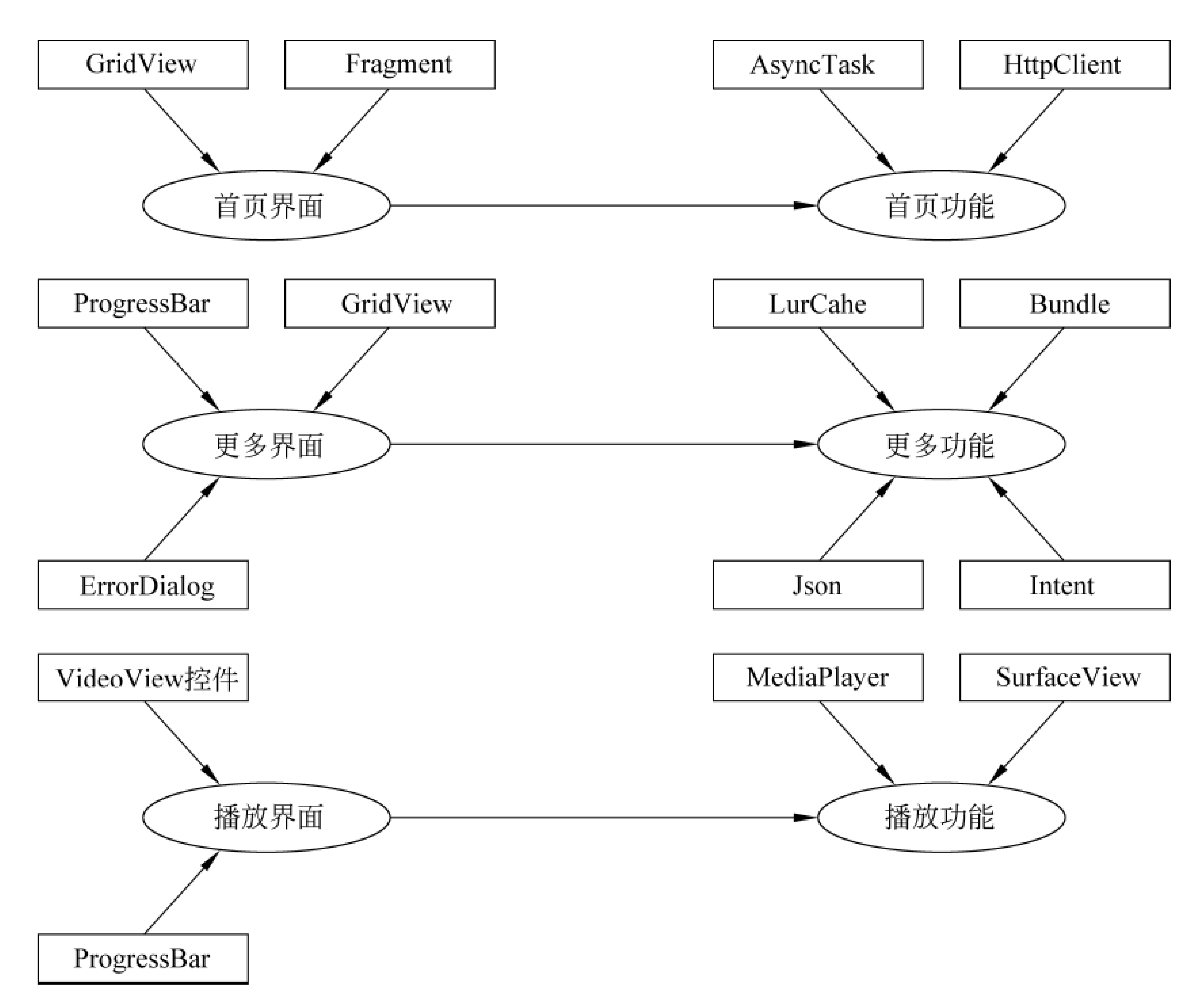


图 8-1 影音播放器的设计及开发

使 ProgressBar 可以在 VideoView 控件上层并居中显示。

4. 播放功能

播放功能使用 VideoView 控件，该控件内部为 MediaPlayer 和 SurfaceView 控件的封装。使用该控件的主要流程为初始化控件—装入 url—设置监听。

8.2 项目界面设计

8.2.1 知识准备

1. Tomcat 配置环境变量的工具/原料

（1）JDK：版本为 jdk-7-windows-i586.exe，下载地址为 http://www.oracle.com/technetwork/java/javase/downloads/index.html。

（2）Tomcat：版本为 apache-tomcat-7.0.33-windows-x86.zip，下载地址为 http://tomcat.apache.org/。

（3）Windows 2003，32bit。

2. Tomcat 配置环境变量的方法/步骤

(1) 安装 JDK 和 Tomcat

① 安装 JDK：直接运行 jdk-7-windows-i586. exe 可执行程序，默认安装。

注：路径可以其他盘符，不建议路径包含中文名及特殊符号。

② 安装 Tomcat：直接解压缩下载文件 apache-tomcat-7. 0. 33-windows-x86. zip 到 C 盘下。安装路径建议修改为：c:\tomcat。

注：如下载的是可执行文件，双击运行，默认安装。

(2) 配置 JDK 环境变量

① 新建变量名：JAVA_HOME，变量值：C:\Program Files\Java\jdk1. 7. 0。

② 打开 PATH，添加变量值：%JAVA_HOME%\bin；%JAVA_HOME%\jre\bin。

③ 新建变量名：CLASSPATH，变量值：.；%JAVA _HOME%\lib\dt. jar；%JAVA _HOME%\lib\tools. jar。

注：

① 表示当前路径，%JAVA_HOME%就是引用前面指定的 JAVA_HOME。

② JAVA_HOME 指明 JDK 安装路径，此路径下包括 lib、bin、jre 等文件夹，tomcat、eclipse 等的运行都需要依靠此变量。

③ PATH 使系统可以在任何路径下识别 java 命令。

④ CLASSPATH 为 java 加载类(class or lib)路径，只有类在 classpath 中，java 命令才能识别。

(3) 测试 JDK

在 CMD 命令下输入 javac、java、javadoc 命令，出现图 8-2 所示界面，表示安装成功。

```
C:\Documents and Settings\Administrator>java -version
java version "1.7.0"
Java(TM) SE Runtime Environment (build 1.7.0-b147)
Java HotSpot(TM) Client VM (build 21.0-b17, mixed mode, sharing)
```

图 8-2　测试 JDK

(4) 配置 Tomcat 环境变量

① 新建变量名：CATALINA_BASE，变量值：C:\tomcat。

② 新建变量名：CATALINA_HOME，变量值：C:\tomcat。

③ 打开 PATH，添加变量值：%CATALINA _HOME%\lib；%CATALINA _HOME%\bin。

(5) 启动 Tomcat 服务

① 方法一：在 CMD 命令下输入命令：startup，出现如下对话框，表明服务启动成功。

② 方法二：右击桌面上的“我的电脑”→“管理”→“服务和应用程序”→“服务”，找到“Apache Tomcat”服务，右击该服务，选择“属性”，将“启动类型”由“手动”改成“自动”。

效果如图 8-3 所示。

```
C:\Documents and Settings\Administrator>startup
Using CATALINA_BASE:   "C:\tomcat"
Using CATALINA_HOME:   "C:\tomcat"
Using CATALINA_TMPDIR: "C:\tomcat\temp"
Using JRE_HOME:        "C:\Program Files\Java\jdk1.7.0"
Using CLASSPATH:       "C:\tomcat\bin\bootstrap.jar;C:\tomcat\bin\tomcat-juli.ja
r"
```

图 8-3　启动 Tomcat 服务

（6）测试 Tomcat

打开浏览器，在地址栏中输入 http://localhost:8080 后按 Enter 键，如果看到 Tomcat 自带的一个 JSP 页面，说明 JDK 和 Tomcat 已搭建成功。

注：

① JAVA_HOME 中的路径不能用分号结尾，如 C:\Program Files\Java\jdk1.7.0。

② CATALINA_BASE、CATALINA_HOME、TOMCAT_HOME 中的路径不能以\结尾。

③ JAVA_HOME 的路径一定不要写成了 JRE 的路径。

④ 在环境变量中修改添加变量时，一定要注意分号、空格，是否有多余的字母。如果配置不成功，一定要反复检查。

以上错误非常容易出现，错误 CATALINA_HOME 或是 JAVA_HOME 没有配置好，如错误则提示 The CATALINA _ HOME environment variable is not defined correctly。

最终效果如图 8-4 所示。

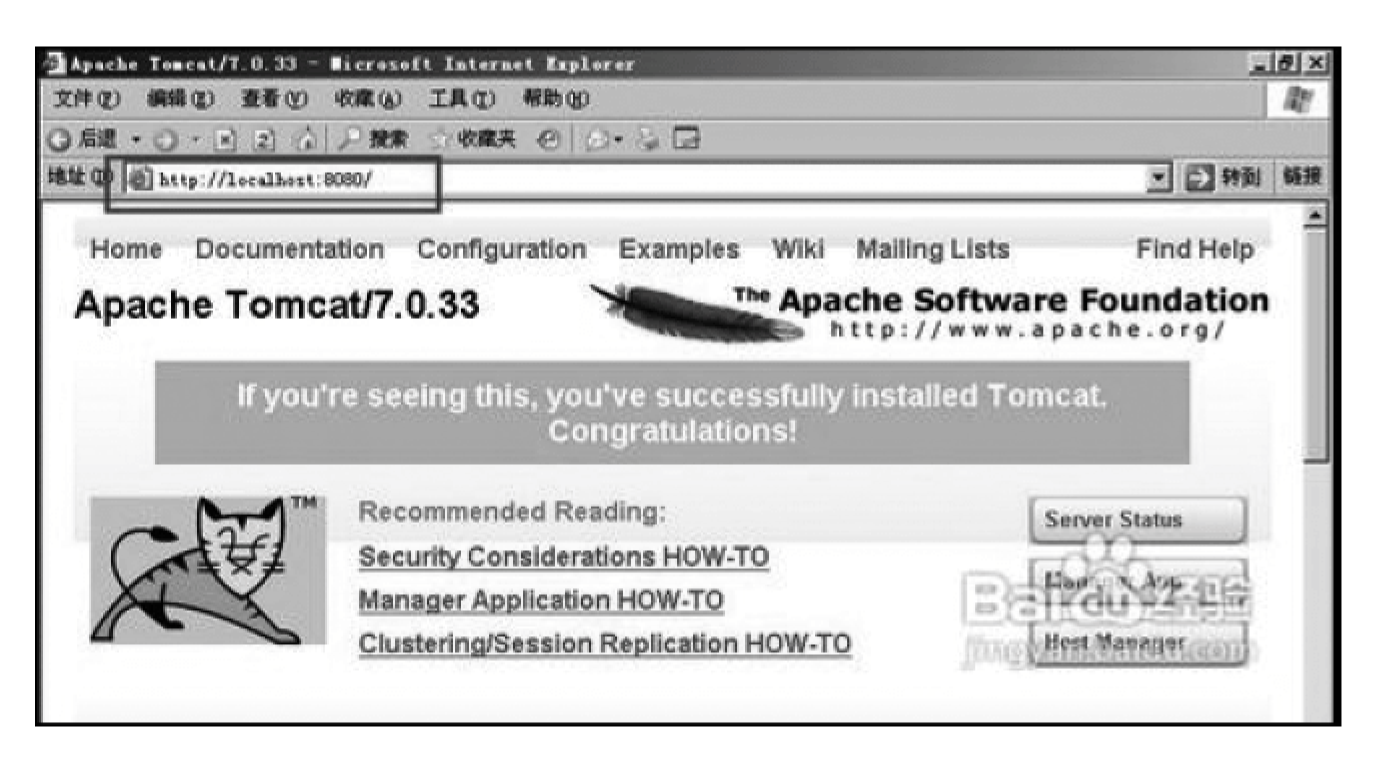


图 8-4　测试 Tomcat

8.2.2　项目界面相关代码设计

（1）主界面设计

主界面框架为标题、底部导航加 Fragment，Fragment 是自定义加载到 LinearLayout 上，所以此处的 LinearLayout 需要有 ID。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="matCH8_parent"
```

```
android:layout_height="matCH8_parent"
tools:context=".MainActivity">

<RelativeLayout
    android:id="@+id/ii_title"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/main_title_bg">

    <ImageView
        android:layout_width="70dip"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_centerVertical="true"
        android:layout_marginLeft="5dip"
        android:src="@drawable/logo" />

    <TextView
        android:id="@+id/ii_title_content"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:textColor="@android:color/black"
        android:textSize="22sp" />

    <ImageView
        android:id="@+id/ii_title_search"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:layout_marginRight="5dip"
        android:background="@drawable/ic_search" />
</RelativeLayout>

<LinearLayout
    android:id="@+id/ii_bottom"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:background="@drawable/main_tab_bg">

    <ImageView
        android:id="@+id/ii_bottom_home"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
```

```
            android:layout_weight="1.0"
            android:src="@drawable/ic_tab_home" />

        <ImageView
            android:id="@+id/ii_bottom_channel"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_weight="1.0"
            android:src="@drawable/ic_tab_channel" />

        <ImageView
            android:id="@+id/ii_bottom_search"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_weight="1.0"
            android:src="@drawable/ic_tab_search" />

        <ImageView
            android:id="@+id/ii_bottom_lottery_myself"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_weight="1.0"
            android:src="@drawable/ic_tab_my" />
    </LinearLayout>

    <RelativeLayout
        android:id="@+id/ii_middle"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_above="@id/ii_bottom"
        android:layout_below="@id/ii_title"
        android:background="@drawable/ic_middle_bg">
    </RelativeLayout>

</RelativeLayout>
```

效果如图 8-5 所示。

(2) 首页 Fargment 布局设计

当前页面为首页的一个 Fragment 子布局，主要包括视频的分类展示。每一个分类下都是使用 GridView（类似九宫格）并且设定了个数。单击更多按钮会跳转到对应分类下的更多界面。单击 GridView 下的每一个 item 返回服务器请求数据，播放视频。

```
<?xml version="1.0" encoding="UTF-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

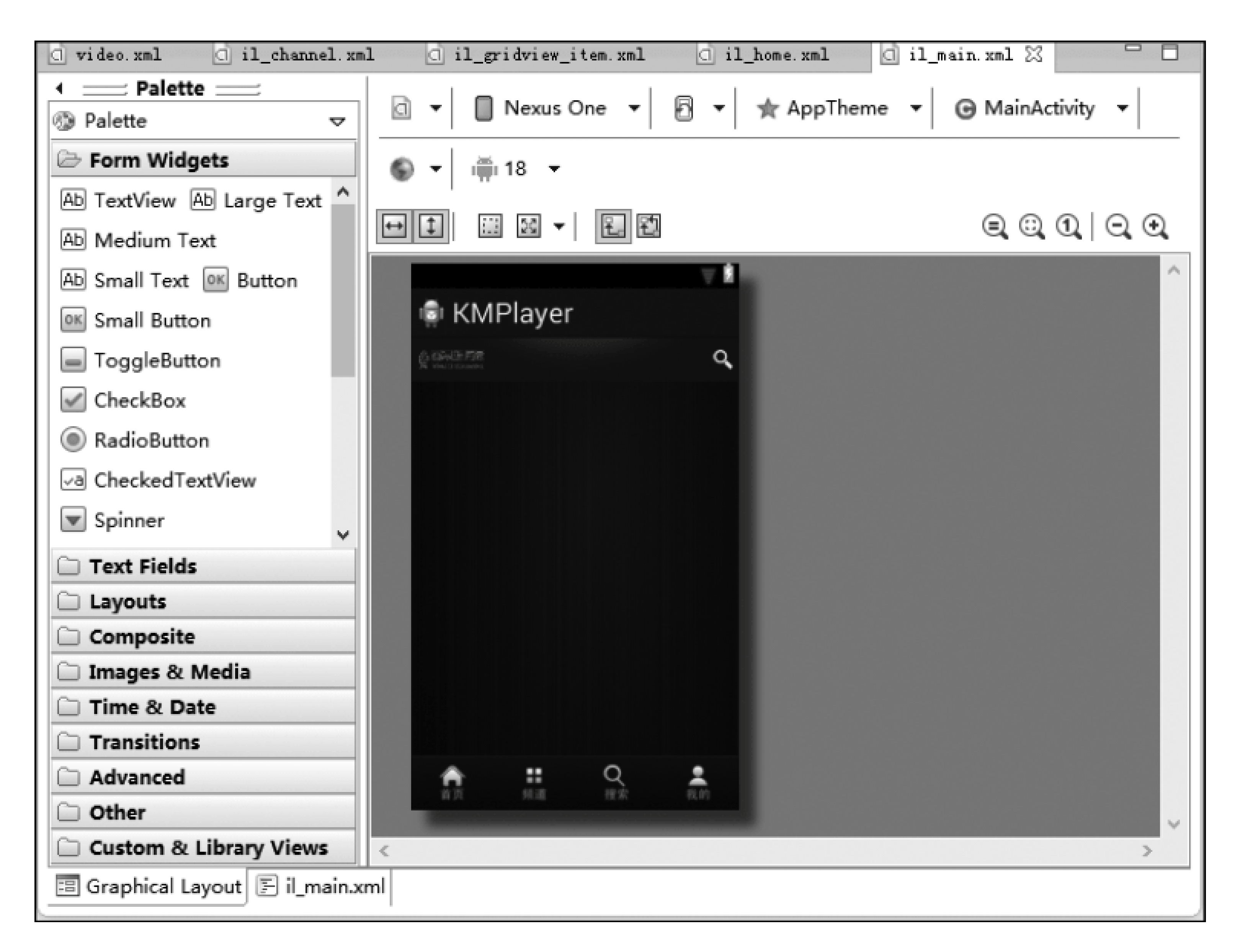


图 8-5　主界面

```
android:layout_width="fill_parent"
android:layout_height="fill_parent">

<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">

    <ScrollView
        android:id="@ id/scroll"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:fadingEdgeLength="0.0dip"
        android:scrollbars="none">

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:orientation="vertical">

            <RelativeLayout
                android:id="@ id/rly_feature_container"
                android:layout_width="fill_parent"
                android:layout_height="160.0dip"
                android:layout_weight="1.21"
                android:background="@ android:color/black"
```

```
    android:orientation="vertical">

    <Gallery
        android:id="@id/newImageGallery"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_alignParentBottom="true"
        android:fadingEdgeLength="0.0dip"
        android:gravity="center_vertical"
        android:unselectedAlpha="1.0" />

    <RadioGroup
        android:id="@id/mainRadioGallery"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:gravity="center"
        android:orientation="horizontal"
        android:paddingLeft="15.0dip" />
</RelativeLayout>

<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/bg_home_title"
    android:gravity="center_vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:paddingLeft="@dimen/title_text_padding_left"
        android:text="【电影】"
        android:textColor="@android:color/white"
        android:textSize="@dimen/common_font_size_18" />

    <TextView
        android:id="@id/recommendMoreMovie"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:gravity="center_vertical"
        android:paddingLeft="@dimen/title_text_padding_left"
        android:text="更多"
        android:textColor="@android:color/white"
        android:textSize="@dimen/common_font_size_18" />
</RelativeLayout>
```

```
<GridView
    android:id="@+id/gridviewMovie"
    android:layout_width="fill_parent"
    android:layout_height="310.0dip"
    android:layout_marginLeft="@dimen/videodetail_padding_
    left"
    android:layout_marginRight="@dimen/videodetail_padding_
    right"
    android:horizontalSpacing="@dimen/videodetail_gridview_
    padding"
    android:numColumns="3"
    android:verticalSpacing="@dimen/videodetail_
    gridview_padding" />

<!--<GridView-->
<!--android:id="@+id/gridviewMovie"-->
<!--android:layout_width="fill_parent"-->
<!--android:layout_height="155dp"-->
<!--android:columnWidth="300dp"-->
<!--android:horizontalSpacing="10dp"-->
<!--android:padding="1dp"-->
<!--android:scrollbars="horizontal"-->
<!--android:stretchMode="spacingWidthUniform"-->
<!--android:verticalSpacing="10dp">-->
<!--</GridView>-->

<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/bg_home_title"
    android:gravity="center_vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:paddingLeft="@dimen/title_text_padding_left"
        android:text="【电视剧】"
        android:textColor="@android:color/white"
        android:textSize="@dimen/common_font_size_18" />

    <TextView
        android:id="@+id/recommendMoreTv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
```

```
        android:gravity="center_vertical"
        android:paddingLeft="@dimen/title_text_padding_left"
        android:text="更多"
        android:textColor="@android:color/white"
        android:textSize="@dimen/common_font_size_18" />
</RelativeLayout>

<GridView
    android:id="@id/gridviewTv"
    android:layout_width="fill_parent"
    android:layout_height="310.0dip"
    android:layout_marginLeft="@dimen/videodetail_padding_
    left"
    android:layout_marginRight="@dimen/videodetail_
    padding_right"
    android:horizontalSpacing="@dimen/videodetail_gridview_
    padding"
    android:numColumns="3"
    android:verticalSpacing="@dimen/videodetail_gridview_
    padding" />

<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/bg_home_title"
    android:gravity="center_vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:paddingLeft="@dimen/title_text_padding_left"
        android:text="【综艺】"
        android:textColor="@android:color/white"
        android:textSize="@dimen/common_font_size_18" />

    <TextView
        android:id="@id/recommendMoreZy"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:gravity="center_vertical"
        android:paddingLeft="@dimen/title_text_padding_left"
        android:text="更多"
        android:textColor="@android:color/white"
        android:textSize="@dimen/common_font_size_18" />
</RelativeLayout>
```

```
<GridView
    android:id="@ id/gridviewZy"
    android:layout_width="fill_parent"
    android:layout_height="310.0dip"
    android:layout_marginLeft="@ dimen/videodetail_padding_
    left"
    android:layout_marginRight="@ dimen/videodetail_padding_
    right"
    android:horizontalSpacing="@ dimen/videodetail_gridview_
    padding"
    android:numColumns="3"
    android:verticalSpacing="@ dimen/videodetail_gridview_
    padding" />

<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@ drawable/bg_home_title"
    android:gravity="center_vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:paddingLeft="@ dimen/title_text_padding_left"
        android:text="【动漫】"
        android:textColor="@ android:color/white"
        android:textSize="@ dimen/common_font_size_18" />

    <TextView
        android:id="@ id/recommendMoreDm"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:gravity="center_vertical"
        android:paddingLeft="@ dimen/title_text_padding_left"
        android:text="更多"
        android:textColor="@ android:color/white"
        android:textSize="@ dimen/common_font_size_18" />
</RelativeLayout>

<GridView
    android:id="@ id/gridviewDm"
    android:layout_width="fill_parent"
    android:layout_height="310.0dip"
    android:layout_marginLeft="@ dimen/videodetail_padding_
```

```
        left"
        android:layout_marginRight="@dimen/videodetail_padding_
        right"
        android:horizontalSpacing="@dimen/videodetail_gridview_
        padding"
        android:numColumns="3"
        android:verticalSpacing="@dimen/videodetail_gridview_
        padding" />

    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/bg_home_title"
        android:gravity="center_vertical">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center_vertical"
            android:paddingLeft="@dimen/title_text_padding_left"
            android:text="【美女】"
            android:textColor="@android:color/white"
            android:textSize="@dimen/common_font_size_18" />

        <TextView
            android:id="@id/recommendMoreWoman"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:gravity="center_vertical"
            android:paddingLeft="@dimen/title_text_padding_left"
            android:text="更多"
            android:textColor="@android:color/white"
            android:textSize="@dimen/common_font_size_18" />
    </RelativeLayout>

    <GridView
        android:id="@id/gridviewWoman"
        android:layout_width="fill_parent"
        android:layout_height="140.0dip"
        android:horizontalSpacing="@dimen/videodetail_gridview_
        padding"
        android:numColumns="2"
        android:verticalSpacing="@dimen/videodetail_gridview_
        padding" />

    <RelativeLayout
```

```
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/bg_home_title"
        android:gravity="center_vertical">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center_vertical"
            android:paddingLeft="@dimen/title_text_padding_left"
            android:text="【音乐】"
            android:textColor="@android:color/white"
            android:textSize="@dimen/common_font_size_18" />

        <TextView
            android:id="@id/recommendMoreMusic"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:gravity="center_vertical"
            android:paddingLeft="@dimen/title_text_padding_left"
            android:text="更多"
            android:textColor="@android:color/white"
            android:textSize="@dimen/common_font_size_18" />
    </RelativeLayout>

    <GridView
        android:id="@id/gridviewMusic"
        android:layout_width="fill_parent"
        android:layout_height="140.0dip"
        android:horizontalSpacing="@dimen/videodetail_gridview_
        padding"
        android:numColumns="2"
        android:verticalSpacing="@dimen/videodetail_gridview_
        padding" />

    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/bg_home_title"
        android:gravity="center_vertical">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center_vertical"
            android:paddingLeft="@dimen/title_text_padding_left"
```

```
            android:text="【搞笑】"
            android:textColor="@android:color/white"
            android:textSize="@dimen/common_font_size_18" />

        <TextView
            android:id="@id/recommendMoreAmuse"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:gravity="center_vertical"
            android:paddingLeft="@dimen/title_text_padding_left"
            android:text="更多"
            android:textColor="@android:color/white"
            android:textSize="@dimen/common_font_size_18" />
    </RelativeLayout>

    <GridView
        android:id="@id/gridviewAmuse"
        android:layout_width="fill_parent"
        android:layout_height="140.0dip"
        android:horizontalSpacing="@dimen/videodetail_
        gridview_padding"
        android:numColumns="2"
        android:verticalSpacing="@dimen/videodetail_gridview_
        padding" />

    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/bg_home_title"
        android:gravity="center_vertical">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center_vertical"
            android:paddingLeft="@dimen/title_text_padding_left"
            android:text="【体育】"
            android:textColor="@android:color/white"
            android:textSize="@dimen/common_font_size_18" />

        <TextView
            android:id="@id/recommendMoreSport"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:gravity="center_vertical"
```

```
            android:paddingLeft="@dimen/title_text_padding_left"
            android:text="更多"
            android:textColor="@android:color/white"
            android:textSize="@dimen/common_font_size_18" />
</RelativeLayout>

<GridView
        android:id="@id/gridviewSport"
        android:layout_width="fill_parent"
        android:layout_height="140.0dip"
        android:horizontalSpacing="@dimen/videodetail_gridview_
        padding"
        android:numColumns="2"
        android:verticalSpacing="@dimen/videodetail_gridview_
        padding" />

<RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/bg_home_title"
        android:gravity="center_vertical">

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center_vertical"
            android:paddingLeft="@dimen/title_text_padding_left"
            android:text="【新闻】"
            android:textColor="@android:color/white"
            android:textSize="@dimen/common_font_size_18" />

        <TextView
            android:id="@id/recommendMoreInfo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:gravity="center_vertical"
            android:paddingLeft="@dimen/title_text_padding_left"
            android:text="更多"
            android:textColor="@android:color/white"
            android:textSize="@dimen/common_font_size_18" />
</RelativeLayout>

<GridView
        android:id="@id/gridviewInfo"
        android:layout_width="fill_parent"
        android:layout_height="140.0dip"
```

```
                android:horizontalSpacing="@dimen/videodetail_gridview_
                padding"
                android:numColumns="2"
                android:verticalSpacing="@dimen/videodetail_gridview_
                padding" />
            </LinearLayout>
        </ScrollView>
    </LinearLayout>

</RelativeLayout>
```

效果如图 8-6 所示。

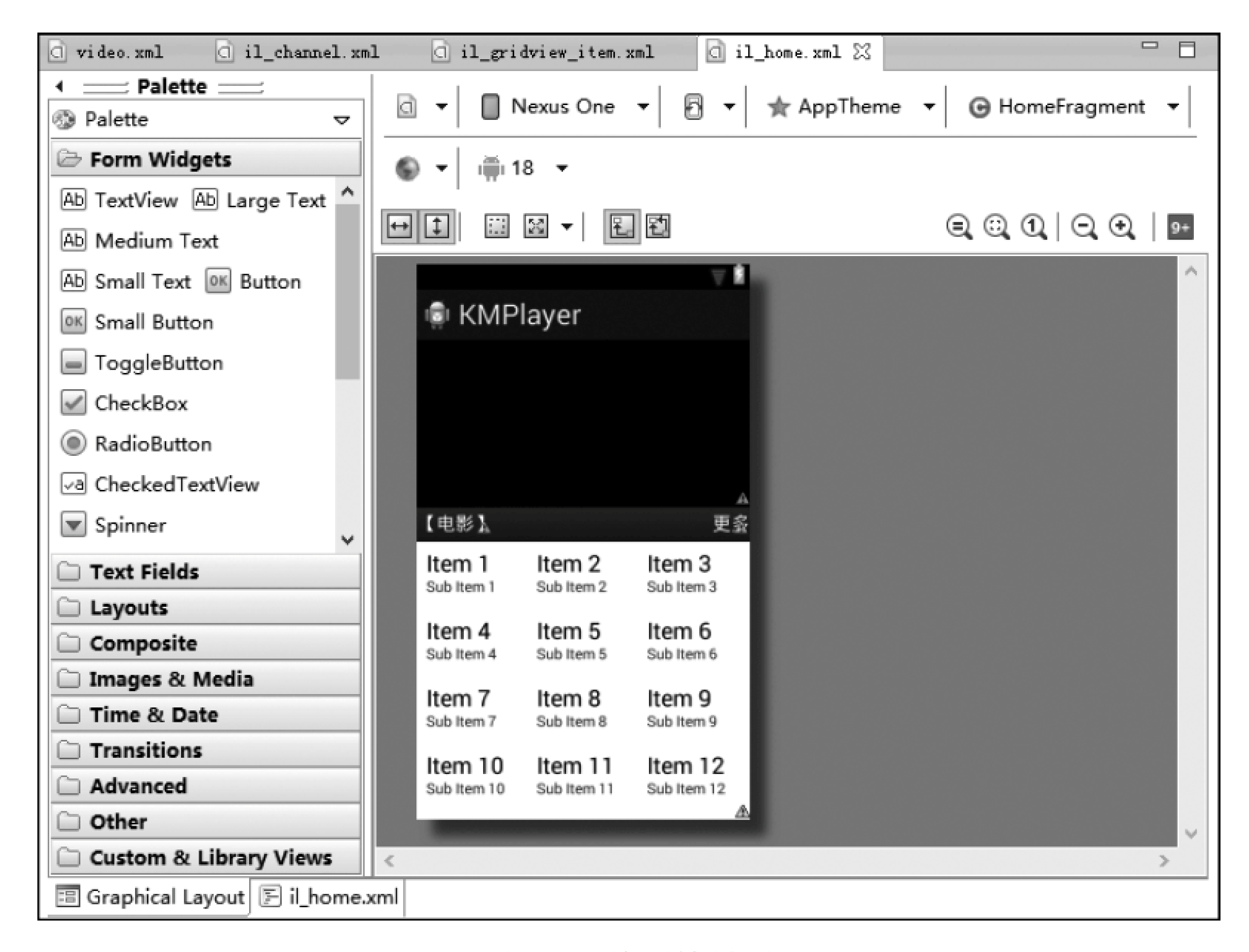


图 8-6　首页效果图

(3) 更多资源界面设计

该界面同为 GridView 控件，并且在底部设置了 ProgressBar 控件，以达到拖动界面加载资源时给予提示的效果。所以 ProgressBar 控件需要在 Java 代码中对其 visibility 属性进行操作，在某特定时刻完成显示和隐藏。

```
<?xml version="1.0" encoding="UTF-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <GridView
        android:id="@id/ii_channle_grid"
```

```
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_gravity="center_horizontal"
        android:layout_marginBottom="15.0dip"
        android:layout_marginLeft="6.0dip"
        android:layout_marginRight="6.0dip"
        android:fadingEdge="none"
        android:fadingEdgeLength="0.0dip"
        android:horizontalSpacing="8.0dip"
        android:numColumns="3"
        android:scrollbars="none"
        android:verticalSpacing="5.0dip" />

    <LinearLayout
        android:id="@ id/progress_linear"
        android:layout_width="wrap_content"
        android:layout_height="30.0dip"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10.0dip"
        android:gravity="center"
        android:orientation="horizontal">

        <ProgressBar
            android:id="@ id/progressbar_loading"
            style="?android:progressBarStyleSmall"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <TextView
            android:id="@ id/load_more_textview"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="5.0dip"
            android:gravity="clip_vertical"
            android:text="@ string/loading"
            android:textColor="@ android:color/white"
            android:textSize="15.0sp" />
    </LinearLayout>

</RelativeLayout>
```

效果如图 8-7 所示。

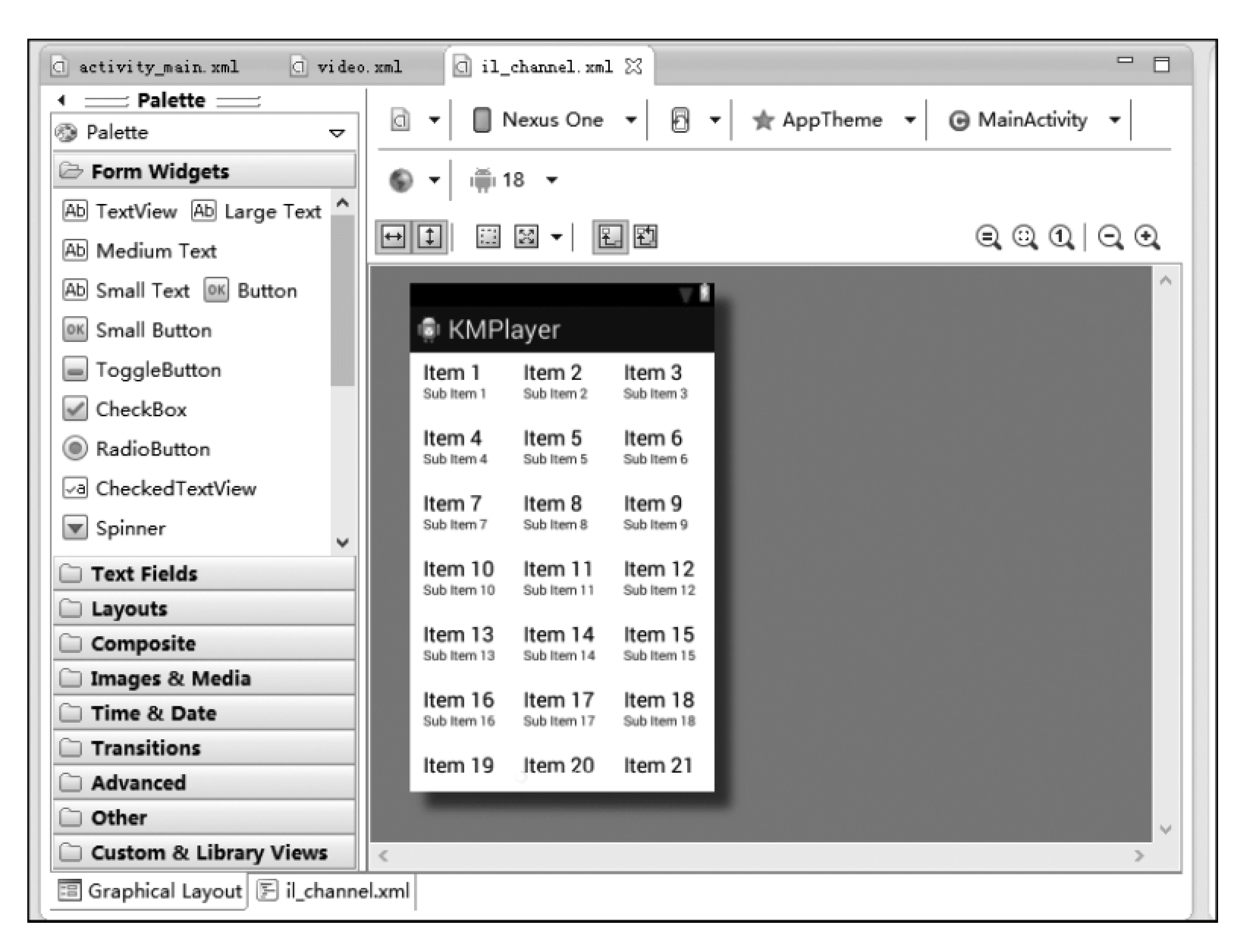


图 8-7 更多界面效果图

（4）GridView 中 Item 设计

```
<?xml version="1.0" encoding="UTF-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <RelativeLayout
        android:id="@ id/lin"
        android:layout_width="100.0dip"
        android:layout_height="130.0dip">

        <ImageView
            android:id="@ id/item_img"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_centerInParent="true"
            android:layout_marginBottom="2.0dip"
            android:layout_marginLeft="2.0dip"
            android:layout_marginRight="2.0dip"
            android:layout_marginTop="2.0dip" />

        <TextView
            android:id="@ id/scoreId"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_marginBottom="5.0dip"
```

```
                android:layout_marginLeft="5.0dip"
                android:singleLine="true"
                android:textColor="@android:color/white"
                android:textSize="14.0sp" />
        </RelativeLayout>

        <TextView
            android:id="@id/txt_loading"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_below="@id/lin"
            android:layout_centerHorizontal="true"
            android:maxWidth="100.0dip"
            android:singleLine="true"
            android:textColor="@android:color/white" />

</RelativeLayout>
```

效果如图 8-8 所示。

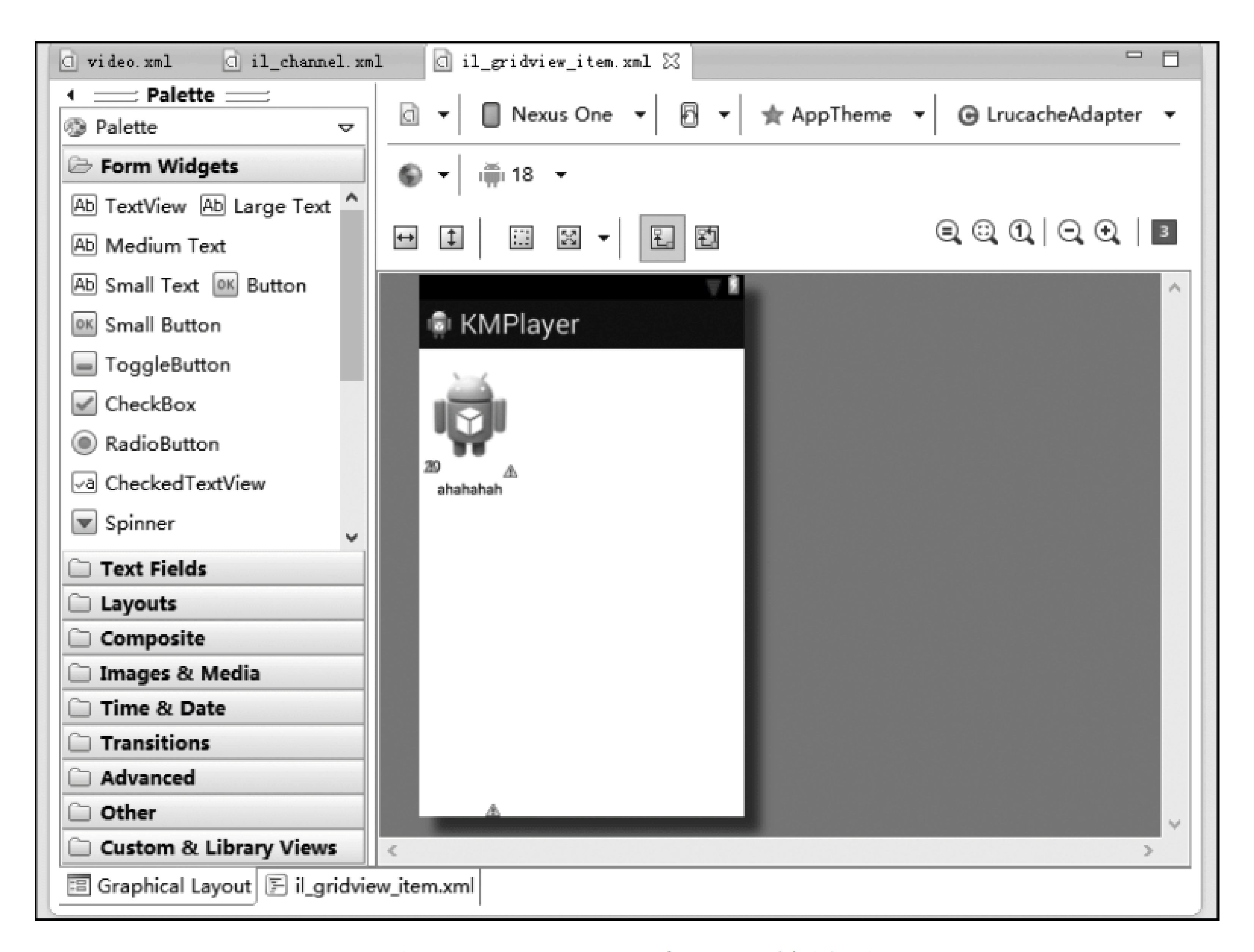


图 8-8　GridView 中 Item 效果图

(5) 播放界面设计

此界面中使用了 VideoView 控件完成视频的播放，加载视频时在 VideoView 上放置 ProgressBar 和 TextView 提示用户。VideoView 控件是 MediaPlayer 和 SurfaceView 的集合，下文中的知识准备模块会详细介绍 MediaPlayer 和 SurfaceView。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="matCH8_parent"
    android:layout_height="matCH8_parent"
    android:orientation="vertical">

    <VideoView
        android:id="@+id/vv"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_centerInParent="true" />

    <LinearLayout
        android:id="@+id/player_loading"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="@drawable/video_player_background"
        android:gravity="center"
        android:orientation="vertical">

        <TextView
            android:id="@+id/loading_video_name"
            android:layout_width="200dip"
            android:layout_height="wrap_content"
            android:ellipsize="start"
            android:gravity="center"
            android:singleLine="true"
            android:text="影音加载..."
            android:textSize="18.6sp" />

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="10.5dip"
            android:gravity="center"
            android:orientation="horizontal">

            <ProgressBar
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" />

            <TextView
                android:id="@+id/loading_text"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginLeft="7.0dip"
                android:textSize="12.6sp" />
        </LinearLayout>
    </LinearLayout>

</RelativeLayout>
```

效果如图 8-9 所示。

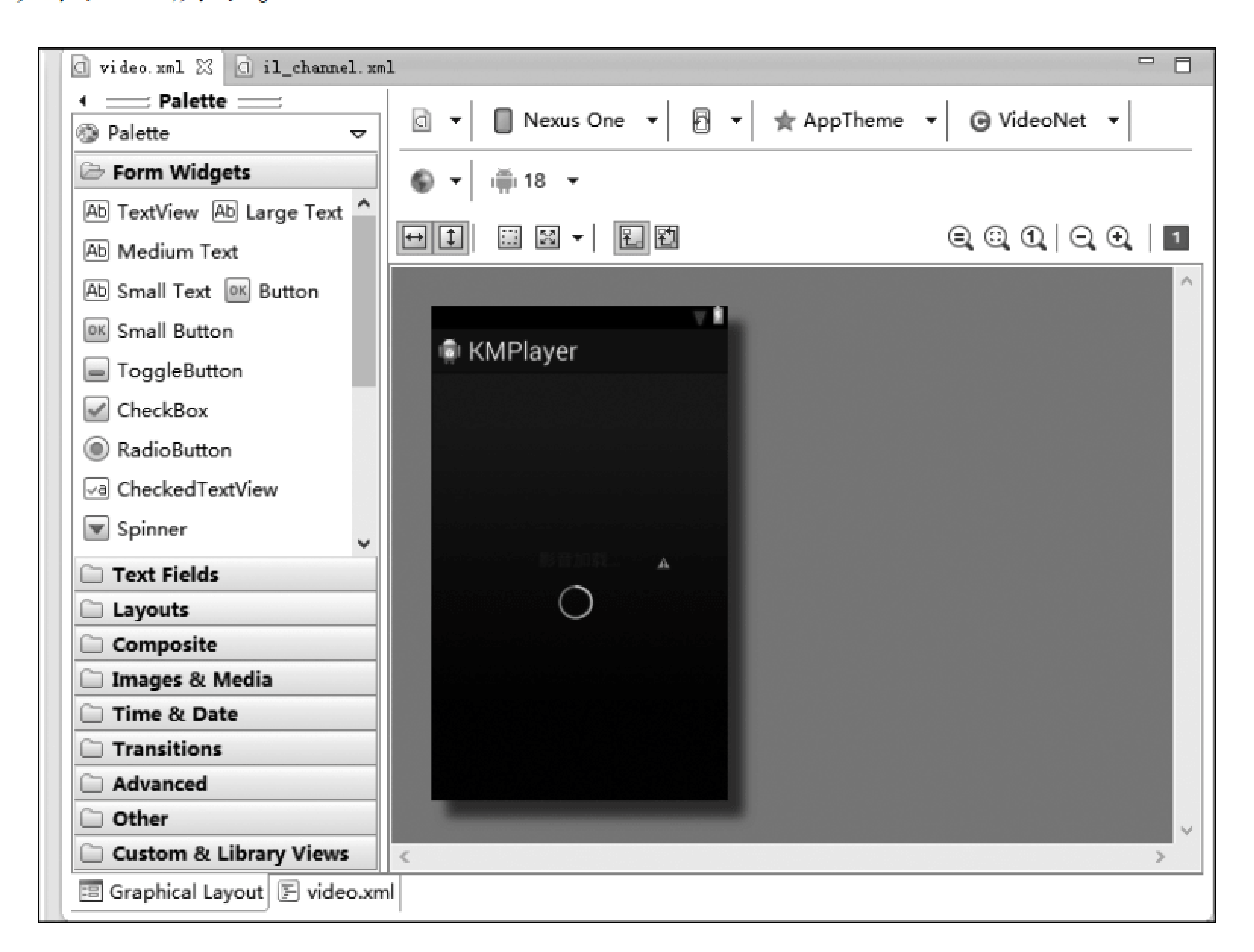


图 8-9　播放界面

8.3　项目功能的实现

8.3.1　知识准备

1. MediaPlayer 的基本用法

(1) 获得 MediaPlayer 实例

① 可以使用直接 new 方式获得 MediaPlayer 实例。

```
MediaPlayer mp= new MediaPlayer();
```

② 也可以使用 create 方式获得 MediaPlayer 实例。

```
MediaPlayer mp= MediaPlayer.create(this, R.raw.test);
//这时就不用调用 setDataSource
```

(2) 设置要播放的文件

MediaPlayer 要播放的文件主要包括以下三个来源。

① 用户在应用中事先自带的 resource 资源。例如：

```
MediaPlayer.create(this, R.raw.test);
```

② 存储在 SD 卡或其他文件路径下的媒体文件。例如：

```
mp.setDataSource("/sdcard/test.mp3");
```

③ 网络上的媒体文件。例如：

```
mp.setDataSource("http://www.citynorth.cn/music/confucius.mp3");
```

MediaPlayer 的 setDataSource 共有以下四个方法：setDataSource（String path）；setDataSource（FileDescriptor fd）；setDataSource（Context context，Uri uri）；setDataSource（FileDescriptor fd，long offset，long length）。

（3）对播放器的主要控制方法

Android 通过控制播放器的状态控制媒体文件的播放，其中，① prepare（）和 prepareAsync()提供了同步和异步两种方式设置播放器进入准备状态。需要注意的是，如果 MediaPlayer 实例是由 create 方法创建的，那么第一次启动播放前不需要再调用 prepare()，因为 create 方法里已经调用过。② start（）是真正启动文件播放的方法。③pause()和 stop()比较简单，起到暂停和停止播放的作用。④seekTo()是定位方法，可以让播放器从指定的位置开始播放，需要注意的是，该方法是异步方法，返回时并不意味着定位完成，尤其是播放的网络文件，真正定位完成时会触发 OnSeekComplete.onSeekComplete()，也可以调用 setOnSeekCompleteListener(OnSeekCompleteListener)设置监听器来处理。⑤release()可以释放播放器占用的资源，一旦确定不再使用播放器时应当尽早调用它释放资源。⑥reset()可以使播放器从 Error 状态中恢复过来，重新回到 Idle 状态。

（4）设置播放器的监听器

MediaPlayer 提供了一些设置不同监听器的方法，对播放器的工作状态进行更好的监听，以期及时处理各种情况。

例如，setOnCompletionListener（MediaPlayer.OnCompletionListener listener）、setOnErrorListener(MediaPlayer.OnErrorListener listener)等，设置播放器时需要考虑播放器可能出现的情况，设置好监听和处理逻辑，以保持播放器的健壮性。

2. SurfaceView 的基本用法

SurfaceView 是视频播放控件，一般与 MediaPlayer 结合使用播放视频。MediaPlayer 也提供了相应的方法设置 SurfaceView 显示图片，只需要为 MediaPlayer 指定 SurfaceView 显示图像。它的完整签名如下：

```
void setDisplay(SurfaceHolder sh)
```

它需要传递一个 SurfaceHolder 对象，SurfaceHolder 可以理解为 SurfaceView 装载需要显示的一帧帧图像的容器，它可以通过 SurfaceHolder.getHolder()方法获得。

使用 MediaPlayer 配合 SurfaceView 播放视频的步骤，与使用 MediaPlayer 播放 MP3 大体一致，只需要额外设置显示的 SurfaceView。

SurfaceView 双缓冲可以理解为有两个线程轮番去解析视频流的帧图像，当一个线程解析完帧图像后，把图像渲染到界面中，同时另一线程开始解析下一帧图像，使两个线程轮番配合解析视频流，以达到流畅播放的效果，如图 8-10 所示。

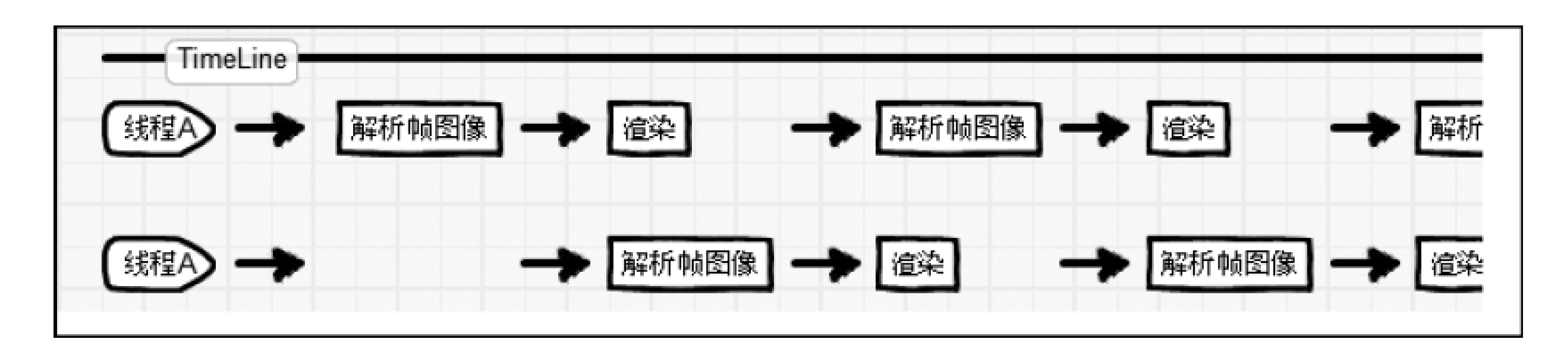


图 8-10 双缓冲

8.3.2 项目功能相关代码设计

1. 主界面逻辑代码实现

(1) 声明两个子布局和底部导航的四个控件以及标题。

```
private HomeFragment homeFragment;
private MoreFragment moreFragment;

private ImageView home;
private ImageView channel;
private ImageView search;
private ImageView myself;

private TextView title;
```

(2) 初始化界面并将界面的上下文存入 GloableParams 类中，以便其他页面的调用。调用初始化控件和监听的方法。

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.il_main);
        GloableParams.MAIN=this;
        init();
        setListener();
    }
```

(3) 初始化两个子布局，将 homeFragment 加入主界面。

```
private void init() {
    moreFragment=new MoreFragment();
    homeFragment=new HomeFragment();
    initBottom();
    addHome();
}
```

(4) 单击事件监听，单击底部导航时改变标题和子布局并且切换底部导航图片，单击更多时加载 moreFragment 布局。

```
@Override
public void onClick(View v) {
    home.setImageResource(getImageId(0,false));
    channel.setImageResource(getImageId(1,false));
    search.setImageResource(getImageId(2,false));
    myself.setImageResource(getImageId(3,false));

    switch (v.getId()) {
    case R.id.ii_bottom_home:
        title.setText("首页");
        home.setImageResource(getImageId(0,true));
        addHome();
        break;

    case R.id.ii_bottom_channel:
        title.setText("频道");
        channel.setImageResource(getImageId(1,true));
        addMore();
        break;

    case R.id.ii_bottom_search:
        title.setText("本地视频");
        search.setImageResource(getImageId(2,true));
        break;

    case R.id.ii_bottom_lottery_myself:
        title.setText("我的影视大全");
        myself.setImageResource(getImageId(3,true));
        break;
    }

}
```

（5）底部导航图片的切换。

```
private int getImageId(int paramInt, boolean paramBoolean) {
    switch (paramInt) {
    case 0:
        if(paramBoolean){
            return R.drawable.ic_tab_home_press;
        }
        return R.drawable.ic_tab_home;

    case 1:
        if(paramBoolean){
            return R.drawable.ic_tab_channel_press;
        }
        return R.drawable.ic_tab_channel;

    case 2:
```

```
            if (paramBoolean) {
                return R.drawable.ic_tab_searCH8_press;
            }
            return R.drawable.ic_tab_search;

        case 3:
            if (paramBoolean) {
                return R.drawable.ic_tab_home_press;
            }
            return R.drawable.ic_tab_my;

        default:
            return-1;
        }
    }
```

（6）播放页面的跳转。

```
public void jumpActivity(String uri) {
    Intent intent=new Intent(this,VideoNet.class);
    intent.putExtra("uri", uri);
    startActivity(intent);
}
```

（7）调用 UIManager 中的 changeFragment 方法将子视图加载进来。UIManager.getInstance()为单例模式，只能有一个对象被创建。

```
private void addMore() {

    UIManager.getInstance().changeFragment(moreFragment,null, true);
}
```

（8）UIManager 中的 changeFragment 方法：参数 1 为视图对象、参数 2 为数据、参数 3 为是否添加到返回栈中。GloableParams. MAIN 为前文添加的上下文，此处通过上下文得到 FragmentManager 对象，使用 transaction 开启事务来添加视图。使用 replace 方法添加视图时会把上一个视图给替换，这样就不会出现两个视图同时出现的 bug。但是这种方法正使用返回键时无法返回上一个视图，所以需要将当前视图手动的加入返回栈中。

```
public void changeFragment(Fragment target,Bundle bundle,boolean isAdd) {
    if (bundle != null) {
        target.setArguments(bundle);
    }
    FragmentManager manager=GloableParams.MAIN.
    getSupportFragmentManager();
    FragmentTransaction transaction=manager.beginTransaction();
    transaction.replace(R.id.ii_middle, target);

    //返回键的操作
```

```
        if(isAdd){
            transaction.addToBackStack(null);        //加入返回键的栈中
        }

        transaction.commit();
    }
```

效果如图 8-11 所示。



图 8-11 主界面

2. 首页逻辑代码实现

(1) 主界面默认加载的是 homeFragment,即首页。在 onCreateView 方法中进行控件和适配器的绑定。

```
@Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
            Bundle savedInstanceState) {
        //getView
        Log.i(TAG,"onCreateView");
        View view= inflater.inflate(R.layout.il_home, container, false);
        //container:LayoutParameter参数传递;attachToRoot:false不会挂载到 root上
        this.gridviewMovie= ((GridView) view.findViewById
        (R.id.gridviewMovie));                //九宫图
```

(2) 初始化适配器,参数中的 6 为图片的个数。

```
adapter= new SoftcacheAdapter(getActivity(), 6);

this.gridviewMovie.setAdapter(adapter);
this.recommendMoreMovie= ((TextView) view
        .findViewById(R.id.recommendMoreMovie));
```

(3) setCallback 方法在从服务器上获取图片数据后，回调将图片资源加载到视图中。

```
adapter.setCallback(new ImageCallback() {        //回调，即绑定监听器

    @Override
    public void imageLoaded(Bitmap bitmap, Object tag) {
        ImageView imageView= (ImageView) gridviewMovie
                .findViewWithTag(tag);

        if (imageView != null) {
            imageView.setImageBitmap(bitmap);
        }
    }
});
```

(4) TextView 控件更多的单击监听。单击后拿到服务器请求的 uri 放入 Bundle 对象中，在加载更多页面时传递过去。

```
recommendMoreMovie.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        Bundle bundle= new Bundle();
        bundle.putString("data", ConstantValue.URI
                + ConstantValue.VIDEO_URI);
        UIManager.getInstance().changeFragment(new MoreFragment(),
                bundle,true);

        }
    });
    return view;
}
```

(5) 拿到服务器请求的 uri，调用 NetWorkTask(). executeProxy 异步方法请求服务器。将监听的上下文赋值给 params. listener，传入调用的方法中。

```
@Override
    public void onStart() {
        Log.i(TAG,"onStart");
        //发送请求，读取 Json 信息
        //发送的参数：①onResultListener 实现类的对象 ②URL ③Map<String,String> prams ④是否弹出
          滚动条
        Params params= new Params();
        params.listener= this;//父类 BaseFragment 实现了此监听
        params.url= ConstantValue.URI+ ConstantValue.SLICE_URI;
        //首页列表的 json

        new NetWorkTask().executeProxy(params);
        super.onStart();
    }
```

(6) 进行网络判断，如果当前网络不正常，调用 PromptManager. showNoNetWork (mContext)；检查是否有网络。若网络状态良好则调用 super. execute(params)进行服务器请求。此过程中弹出进度条对话框提示资源加载中。

在此方法中，将之前设置的上下文赋给 this. onResultListener 就可以在当前页面调用首页面。

```
/**
 * 加强版的开始线程的操作(网络判断)
 *
 * @param params
 *            <p>
 *            <li> index=0 的参数为 Fragment
 *            <li> index=1 的参数为是否显示滚动条
 *            <li> index=2 链接
 * @return
 * /
public final AsyncTask<Params, Integer, Object> executeProxy(
        Params... params) {
    if (params[0].listener instanceof BaseFragment) {
        this.onResultListener=params[0].listener;
        mContext=((BaseFragment) params[0].listener).getActivity();
        //判断网络的状态
        if (NetUtil.checkNetType(mContext)) {
            if ((Boolean) params[0].isShowProgress) {
                PromptManager.showProgressDialog(mContext);    //弹出对话框
            }
            return   super.execute(params);                //调用 onPostExecute
        } else {
            PromptManager.showNoNetWork(mContext);         //检查是否有网络
        }
    }
    return null;
}
```

(7) super. execute(params)会调用 doInBackground 方法，此方法中调用 clientUtil. sendGet 方法请求服务器。

```
@Override
    protected Object doInBackground(Params... params) {
        HttpClientUtil clientUtil=new HttpClientUtil();
        String result=clientUtil.sendGet(params[0].url);
        return result;
    }
```

(8) 使用 HttpGet 请求并加入 get. setParams(httpParams)对请求进行一定的设置。如果请求成功，则将得到的流对象转换成字符串返回，否则返回 null。

```
public String sendGet(String uri) {                        //请求 json 数据
    get=new HttpGet(uri);
```

```
            HttpParams httpParams=new BasicHttpParams();          //请求参数
            HttpConnectionParams.setConnectionTimeout(httpParams, 8000);
                                                                  //设置连接超时时间
            HttpConnectionParams.setSoTimeout(httpParams, 8000);                  //响应超时
            get.setParams(httpParams);

            try {
                response=client.execute(get);

                if (response.getStatusLine().getStatusCode()==200) {
                    return EntityUtils.toString(response.getEntity(), ConstantValue.ENCODING);
                }
            } catch (Exception e) {
                e.printStackTrace();
            }

            return null;
        }
```

(9) doInBackground 方法结束后会调用 onPostExecute 得到返回值，并关闭进度条对话框。如果当前 onResultListener 对象不为 null，则根据返回的 result 设置 errorCode，并且调用 onResultListener. onGetResult 方法结果和返回码返回。

```
        /**
         * 请求的返回结果
         */
        protected void onPostExecute(Object result) {
            PromptManager.closeProgressDialog();
            if (this.onResultListener != null) {
                int errorCode=ConstantValue.SUCCESS;
                if (result==null) {
                    errorCode=ConstantValue.ERROR;
                }

                onResultListener
                        .onGetResult(errorCode==ConstantValue.ERROR ? null
                                : result, errorCode);       //将结果和返回码返回
            }
        }
```

(10) 对返回的结果进行判断，如果返回值不为空并且返回码为 ConstantValue. SUCCESS，那么就将 result 所对应的字符串解析出来(JSON)，将其中的 uri 放入集合并将集合传入 adapter，通知 adapter 更新界面；否则弹出对话框提示。

```
        @Override
        public void onGetResult(Object result, int iError) {
            if(iError==ConstantValue.SUCCESS && result != null){
                                                        //获取返回的json信息并解析使用
```

```
            String json= result.toString();
            try{
                JSONObject jsonObject= new JSONObject(json);
                List< Slice> parseArray= JSON.parseArray(jsonObject.getString
                ("slices"), Slice.class);
                Slice slice= parseArray.get(0);
                List< Hot> hot= slice.getHot();

                //更新界面
                adapter.setHotList(hot);
                adapter.notifyDataSetChanged();
            }catch(Exception e){

            }
        }else{
            //getActivity() 当前系统仅有的 Activity的引用
            PromptManager.showErrorDialog(getActivity(), "服务器忙 ......");
        }
        super.onGetResult(result, iError);
}
```

(11) 在适配器创建时就绑定回调函数。

```
public void setCallback(ImageCallback callback) {
    this.callback= callback;
}
```

(12) 为控件绑定数据。

```
@Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder holder= null;
    }
```

(13) 判断是否存在 convertView,若存在直接拿来重用,若不存在则创建一个。

```
if (convertView != null) {
    holder= (ViewHolder) convertView.getTag();
} else {
    holder= new ViewHolder();
    convertView= View
            .inflate(context, R.layout.il_gridview_item, null);

    holder.itemImg= (ImageView) convertView
            .findViewById(R.id.item_img);
    holder.scord= (TextView) convertView.findViewById(R.id.scoreId);
    holder.title= (TextView) convertView
            .findViewById(R.id.txt_loading);

    holder.itemImg.setImageResource(R.drawable.id_default);
```

```
        convertView.setTag(holder);
    }

    if (hotList != null) {
        try {
```

(14) 拿到对应的 hot 对象，该对象中包括 uri 等视频详细信息。

```
final Hot hot=hotList.get(position);
```

(15) item 的单击事件，使用上下文调用主界面的方法带着 uri 跳转到播放界面。

```
convertView.setOnClickListener(new OnClickListener() {
                                    //单击 item 跳转到视频播放页面,并传递指定的 uri

    @Override
    public void onClick(View arg0) {
        ((MainActivity)GloableParams.MAIN).jumpActivity(hot.getUrl());

    }
});

if (hot != null) {
```

(16) 拿到图片的 uri。

```
String imgUrl=hot.getImg_url();
if (StringUtils.isNotBlank(imgUrl)) {
                    //判断某字符串是否不为空且长度不为 0 且不由空白符(whitespace)构成
```

(17) 将 getWorks_id 赋给图片让每一张图片都有唯一的 tag。

```
//tag 的指定唯一
holder.itemImg.setTag(hot.getWorks_id());
```

(18) 在软引用中通过唯一的 tag 找寻这张图片，如果存在就赋值给 itemImg，否则判断 callback 是否存在对象(因为在一开始就设置了回调所以该对象不为 null)，存在就调用 ImageDownload(callback).execute 方法，通过 uri 从服务器拿数据并将 callback 传过去方便使用回调函数。

```
        Bitmap bm=GloableParams.IMGCACHE.get(hot
                .getWorks_id());
        if (bm != null) {
            holder.itemImg.setImageBitmap(bm);
        } else {
            if (callback != null) {
                new ImageDownload(callback).execute(imgUrl,
                        hot.getWorks_id());
            }
        }
    }
```

```
            }

            holder.scord.setText(hot.getRating()+"分");
            holder.title.setText(hot.getTitle());
        } catch (Exception e) {
        }
    }

    return convertView;
}
```

(19) 如果上述方法在软引用中找不到需要的图片就会调用到 ImageDownload 中的 doInBackground 方法，该方法通过调用 loadImageFromUrl 方法获取图片，并且对图片进行一定的压缩处理。因为此方法在更多界面也会调用到并且两个界面图片所进行的操作不一样，所以需要通过参数进行判断。根据 flag 的值判断应该进行哪种操作。当前该方法将获取的图片和传递过来的 tag 存入软引用中方便下次使用时读取。

```
@Override
    protected Bitmap doInBackground(String... params) {
        Bitmap bitmap=loadImageFromUrl(params[0]);
        if (bitmap != null) {
            tag=params[1];
            String flag=CACHE_TYPE_SOFT;
            try {
                flag=params[2];
            } catch (Exception e) {
            }
            if (CACHE_TYPE_LRU.equals(flag)) {
                ImageCache.getInstance().put(tag, bitmap);
            } else if (CACHE_TYPE_DIS.equals(flag)) {

            } else {
                GloableParams.IMGCACHE.put(tag, bitmap);
            }

        }
        return bitmap;
    }
```

(20) 通过 adapter.loadImg 获取服务器上的图片信息并对其进行处理，详细流程参见方法注释。adapter.loadImg 方法已经在上文中介绍过，此处不再赘述，需要注意的是此方法为上文方法的重载，返回值类型是不一样的。

```
public static Bitmap loadImageFromUrl(String url) {
    //url=StringUtils.replace(url, "10.0.2.2", "192.168.1.101");
    InputStream i=null;
    try {
```

```
            HttpClientUtil adapter= new HttpClientUtil();
            i= adapter.loadImg(url);

            //http://blog.csdn.net/xianming01/article/details/8280434
            BitmapFactory.Options opt= new BitmapFactory.Options();  //对图片进行操作以保证内存空间
            opt.inPreferredConfig= Bitmap.Config.RGB_565;
                                                   //RGB_565 代表 8 位 RGB 位图
            opt.inPurgeable= true;
            //当 isPurgable 设为 true 时,系统中内存不足时,可以回收部分 Bitmap 占据的内存空间,这
              时一般不会出现 OutOfMemory 错误
            opt.inInputShareable= true;
                                  //决定位图是否能够共享一个指向数据源的引用,或者是复制一份

            return BitmapFactory.decodeStream(i, null, opt);              //返回重新生成的图片
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }
```

上文的 doInBackground 方法结束后会调用 onPostExecute 方法，因为前文一层层的传递已经将 imageCallback 对象传了过来，所以在当前页面上可以调用 imageCallback.imageLoaded，将图片加载到界面上。因为 imageLoaded 方法是在首页面重写的，会回到首页调用，所以该方法称为回调。imageLoaded 方法详情参见上文的绑定回调。

```
@Override
    protected void onPostExecute(Bitmap result) {
        if (imageCallback != null)
            imageCallback.imageLoaded(result, tag);
        super.onPostExecute(result);
    }
```

效果如图 8-12 所示。

3．更多界面逻辑代码实现

单击更多后调用 recommendMoreMovie. setOnClickListener 中的 onclick 方法带着 bundle 等参数将 moreFragment，加载到主界面，因为用的是 resplaces，所以首页面会被替换。

（1）隐藏进度条提示框。

```
progressLinear.setVisibility(View.INVISIBLE);        //正在加载提示
```

（2）在 onCreateView 方法中绑定回调。

```
adapter= new LrucacheAdapter(getActivity(),new ImageCallback() {

    @Override
    public void imageLoaded(Bitmap bitmap, Object tag) {
```



图 8-12　首页面的加载

```
                ImageView imageView= (ImageView) channelGridView
                        .findViewWithTag(tag);
                if (imageView != null) {
                    imageView.setImageBitmap(bitmap);
                }
            }
        });
```

（3）得到传递过来的 bundle，获取其中的信息，调用 executeProxy 拿到数据。

```
Bundle bundle= getArguments();
    if (bundle != null) {
        String url= bundle.getString("data");
        Params params= new Params();
        params.isShowProgress= true;
        params.listener= this;
        params.url= url;

        new NetWorkTask().executeProxy(params);
    }
```

（4）调用该方法得到 result 和 iError，并隐藏进度条。如果 adapter 中已经有视频的详细信息数据，就将请求返回的数据集合加入进去；如果没有，将数据集合设置到 adapter 中，通知界面更新。

```
@Override
    public void onGetResult(Object result, int iError) {
        progressLinear.setVisibility(View.INVISIBLE);
        if (iError= =ConstantValue.SUCCESS && result != null) {
            String json= result.toString();
            try {
```

```
                JSONObject jsonObject= new JSONObject(json);
                List<Video> videos= JSON.parseArray(
                        jsonObject.getJSONObject("video_list").getString(
                                "videos"), Video.class);
                //设置 Adappter 更新 GridView
                List<Video> videos2= adapter.getVideos();
                if (videos2 != null && videos2.size()> 0) {
                    videos2.addAll(videos);
                    adapter.setVideos(videos2);
                } else {
                    adapter.setVideos(videos);
                }
                adapter.notifyDataSetChanged();
            } catch (JSONException e) {
                e.printStackTrace();
            }
        } else {
            //getActivity() 当前系统仅有的 Activity的引用
            PromptManager.showErrorDialog(getActivity(), "服务器忙……");
        }
        super.onGetResult(result, iError);
    }
```

(5) Adapter 的构造中也将回调对象传进来。

```
public LrucacheAdapter(Context context, ImageCallback callback) {
    super();
    this.context= context;
    this.callback= callback;
}
```

(6) getView 方法中进行数据的查询适配。

```
if (videos != null) {
            try {
                Video video= videos.get(position);
                if (video != null) {
                    String imgUrl= video.getImg_url();
                    if (StringUtils.isNotBlank(imgUrl)) {
                        holder.itemImg.setTag(video.getWorks_id());
```

(7) 获取强引用中的数据。

```
Bitmap bm= ImageCache.getInstance().get(video.getWorks_id());
Bitmap bm= GloableParams.IMGCACHE.get(video.getWorks_id());

if (bm != null) {
    holder.itemImg.setImageBitmap(bm);
} else {
    holder.itemImg.setImageResource(R.drawable.id_default);
    if (callback != null) {
```

请求服务器获取图片信息，ImageDownload. CACHE_TYPE_LRU 的作用是将图片放入强引用中。

```
                                new ImageDownload(callback).execute(imgUrl, video.getWorks_id(),
                                ImageDownload.CACHE_TYPE_LRU);
                            }
                        }
                    }
                }

                holder.scord.setText(video.getRating()+"分");
                holder.title.setText(video.getTitle());
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
```

(8) 请求结束后回调该方法，将图片加载到界面上，在获得图片和加载之前会将图片存入强引用中，下次使用该图片会先去强引用中取。详细步骤同上文的存入弱引用。

```
adapter=new LrucacheAdapter(getActivity(),new ImageCallback() {

    @Override
    public void imageLoaded(Bitmap bitmap, Object tag) {
        ImageView imageView= (ImageView) channelGridView
                .findViewWithTag(tag);
        if (imageView !=null) {
            imageView.setImageBitmap(bitmap);
        }
    }
});
```

(9) 对于滚动的监听，在滚动结束后，判断当前位置进行一系列操作。如果最后的 item 显示在界面上，就要请求服务器获取视频信息，之后的操作同上文的服务器请求步骤。

```
private void setListener() {
    channelGridView.setOnScrollListener(new OnScrollListener() {

        @Override
        public void onScrollStateChanged(AbsListView view, int scrollState) {
            //判断 GridView的滚动状态
            //当 GridView停止滚动
            //获取当前正在显示的最后那个 item的 position信息
            //获取 GridView的所有的 size的总量,集合的 size
            //position==size-1,翻页操作处理
            if (scrollState==SCROLL_STATE_IDLE) {
                if (view.getLastVisiblePosition()==view.getCount()-1) {
                    //获取下一页的资源
```

```
                currentpage++;
                //提示信息的处理
                progressLinear.setVisibility(View.VISIBLE);

                //设置参数
                Params params=new Params();
                params.isShowProgress=false;
                params.listener=MoreFragment.this;
```

（10）根据滚动的次数查询相应的数据，此操作需要访问百度的服务器，因此只有在有网络的情况下才能进行。

```
                params.url="http://app.video.baidu.com/adnativemovie/?beg="
                        +(currentpage * 20)
                        +"&end="
                        +(currentpage * 20+20);

                new NetWorkTask().executeProxy(params);

            }
        }

    }

    @Override
    public void onScroll(AbsListView view, int firstVisibleItem,
            int visibleItemCount, int totalItemCount) {
        //TODO Auto-generated method stub

    }
});

}
```

效果如图8-13和图8-14所示。

4. 视频播放界面逻辑代码实现

（1）单击首页面上的任意item会进入视频播放界面。该界面使用videoView播放视频，videoView内部对Mediaplayer和SurfaceView进行了封装。

```
private void init() {
        Intent intent=getIntent();
        String uriStr=intent.getStringExtra("uri");
```

（2）如果传递过来的信息中没有uri，就播放默认视频。

```
if(StringUtils.isBlank(uriStr)){
    uri=Uri.parse("http://10.0.2.2/KMPlayerService/res/movie/6.mp4");
}else{
```



图 8-13　下拉请求服务器



图 8-14　服务器请求失败

```
    uri=Uri.parse(uriStr);
}

videoView=(VideoView) findViewById(R.id.vv);
preLoad=(LinearLayout) findViewById(R.id.player_loading);
```

(3) 设置 uri。

```
videoView.setVideoURI(uri);
```

(4) 准备完毕的监听，调用 start 方法开始播放视频。

```
videoView.setOnPreparedListener(new OnPreparedListener() {

    @Override
    public void onPrepared(MediaPlayer mp) {
        preLoad.setVisibility(View.INVISIBLE);
        videoView.start();
    }
});
```

（5）如果视频播放完毕就结束此页面，回到调用的页面。

```
/**
    * 视频播放完毕的监听
    * /
      videoView.setOnCompletionListener(new MediaPlayer.OnCompletionListener()
      {
      @Override
      public void onCompletion(MediaPlayer mp)
      {
            finish();
      }
    });
}
```

（6）是否开启对视频的控制，若为 true 则会调用内部布局弹出进度条、暂停或播放和快进快退等操作按钮，无须自行对视频播放控制进行代码编写，极大地减少了代码量。

```
videoView.setMediaController(new MediaController(this));
```

效果如图 8-15 所示。



图 8-15 视频加载中

5. 对于图片引用的逻辑代码实现

通过以下方法进行的图片引用,可以保证在加载大量资源数据的情况下使系统流畅运行,不会导致 OOM。

(1) 设定结合中所能存储数据的最大内存容量和 SD 卡的空间。

```
private static final int MAXSIZE= 1024 * 1024 * 5;//5M
private static final int DIS_CACHE_SIZE= 1024 * 1024 * 10;          //10MB 占据 SDCard的空间
```

(2) 声明两者的对象。

```
private LruCache<Object, Bitmap> lrucache;  //图片的缓存;设置的 Value必须能够计算出所占有的内
                                              存的大小
private DiskLruCache diskLruCache;          //将图片缓存在 SDCard上
```

(3) 构造 private 类型以使用单例。在加入元素时需要能够获得其所占内存的大小,否则无法使用。

```
private ImageCache() {
    //put(key ,object) size+ 1- -集合中所存元素的个数
    //当向集合中添加一个 Bitmap时,能够获知 Bitmap所占用的内存大小
    //maxSize代表当前分配给 Bitmap的集合的内存大小

    lrucache= new LruCache<Object, Bitmap> (MAXSIZE) {

        @Override
        protected int sizeOf(Object key, Bitmap value) {
            //每添加一张图片,size的变动代表该 bitmap占用的内存大小
            //Log.i(TAG, key.toString());
            return getSize(value);
        }
```

(4) 如果加入元素后超过最大的存储空间,则需要清理不常用的元素。

```
    @Override
    protected void entryRemoved(boolean evicted, Object key,
            Bitmap oldValue, Bitmap newValue) {
        //evicted 如果为 True所代表:MAXSIZE不够用
        if (evicted) {
            //MAXSIZE不够用
            Log.i(TAG, "remove:"+ key.toString());
            //两种缓存的结合使用
            //如果内存空间充足,可以将被清除的 bitmap对象存入软引用的集合
            //GloableParams.IMGCACHE.put(key, oldValue);
        }
        super.entryRemoved(evicted, key, oldValue, newValue);
    }

};          //就只能放 5M的图片
```

(5) 如果当前存在SD卡，则在SD卡上开辟出一部分空间存放元素，这样可以在读取不到该引用中的元素时在SD卡上读取，使页面资源加载更加流畅。

```
    if (Environment.getExternalStorageState().equals(
            Environment.MEDIA_MOUNTED)) {
        File externalStorageDirectory= Environment
                .getExternalStorageDirectory();
        String path= externalStorageDirectory.getAbsolutePath()
                + ConstantValue.IMAGE_PATH;
        diskLruCache= DiskLruCache.openCache(GloableParams.MAIN, new File(
                path), DIS_CACHE_SIZE);
    }

}
```

(6) 使用get方法读取元素。

```
public Bitmap get(Object key) {
    //获取硬引用图片
    Bitmap bitmap= lrucache.get(key);

    //如果没有
    if (bitmap==null) {
        //读取 SDCard 中的资源
        if (diskLruCache != null) {
            bitmap= diskLruCache.get(key.toString());
        }
    }

    return bitmap;
}
```

(7) 使用Put方法添加元素，并且同时添加SD卡上。一旦超过最大内存，调用上文的方法清除数据。

```
public void put(Object key, Bitmap value) {
    //如果一旦添加的 Bitmap 所占据的总的内存超过了 MAXSIZE,移除一部分不常使用的 map
    lrucache.put(key, value);

    if (diskLruCache != null) {
        diskLruCache.put(key.toString(), value);
    }
}
```

(8) 清除SD卡和引用集合中的数据。

```
public void clear() {
    lrucache.evictAll();
    if (diskLruCache != null) {
        diskLruCache.clearCache();
```

```
        }

}
```

8.4　系统运行与效果测试

系统运行效果如图 8-16～图 8-18 所示。



图 8-16　首页 1



图 8-17　首页 2

本程序仅对网络视频播放进行了按钮的实现，即视频资源信息的获取和视频播放（videoView 仅支持少部分格式的视频播放，若想播放其他格式的视频，需调用第三方的



图 8-18　播放界面

视频播放软件)。页面中的频道、搜索和我的视频部分有待后期拓展。

8.5　本章小结

通过本章的学习,对影音播放控件 MediaPlayer 和视图绘制控件 SurfaceView 有了初步的了解,能够使用网络请求获取服务器资源并且完成播放功能。MediaPlayer 控件支持的播放格式是很有限的,如果需要播放相对丰富的格式的视频,就需要在项目中引用第三方的控件。

本章中,还接触到 Android 开发中常见的一个问题 OOM,即内存溢出。Android 对每一个应用所分配的内存都是有上限的,如果项目中的某些对象占用了较大的内存又不能及时得到销毁,就会导致内存泄漏,最终导致 OOM。本章中容易导致 OOM 的主要因素在于图片加载。为防止 OOM 的发生,我们在项目中使用了缓存:内存缓存和 SD 卡缓存。这两者相结合将大大提高图片加载的效率,并且减少 OOM 的产生。

8.6　项目实践

(1) 将 APP 安装到实验用手机测试本章的功能。
(2) 更换不同屏幕的模拟器测试显示效果。
(3) 跟换不同版本的模拟器测试功能效果。
(4) 使用不同格式的视频测试播放功能。

第 9 章

基于百度地图的 GPS 设计及开发

本章的工作目标如下：

（1）注册百度地图开发者账号，并创建自己的标度地图应用。

（2）在手机或者模拟器上展示百度地图，实现交通地图和卫星地图的切换。

（3）实现在地图上添加文字、图形覆盖物，为覆盖物设置单击事件。

（4）实现兴趣点的检索和分页查询功能。

（5）实现两地之间的驾车、公交和步行的路径检索。

（6）实现定位功能，在地图上实时地显示自己的位置。

（7）完成程序运行和效果测试。

9.1 项目分析

本项目的学习和操作主要关注以下几点。

（1）了解第三方平台应用集成的大致步骤。

（2）熟悉地图控件的使用和该控件的基本属性方法。

（3）掌握对于地图 SDK 的初始化流程、各类的方法和作用以及它们之间的依赖关系。

（4）综合上述的几点，完成本章应用的开发。

基于百度地图的 GPS 设计及开发如图 9-1 所示。

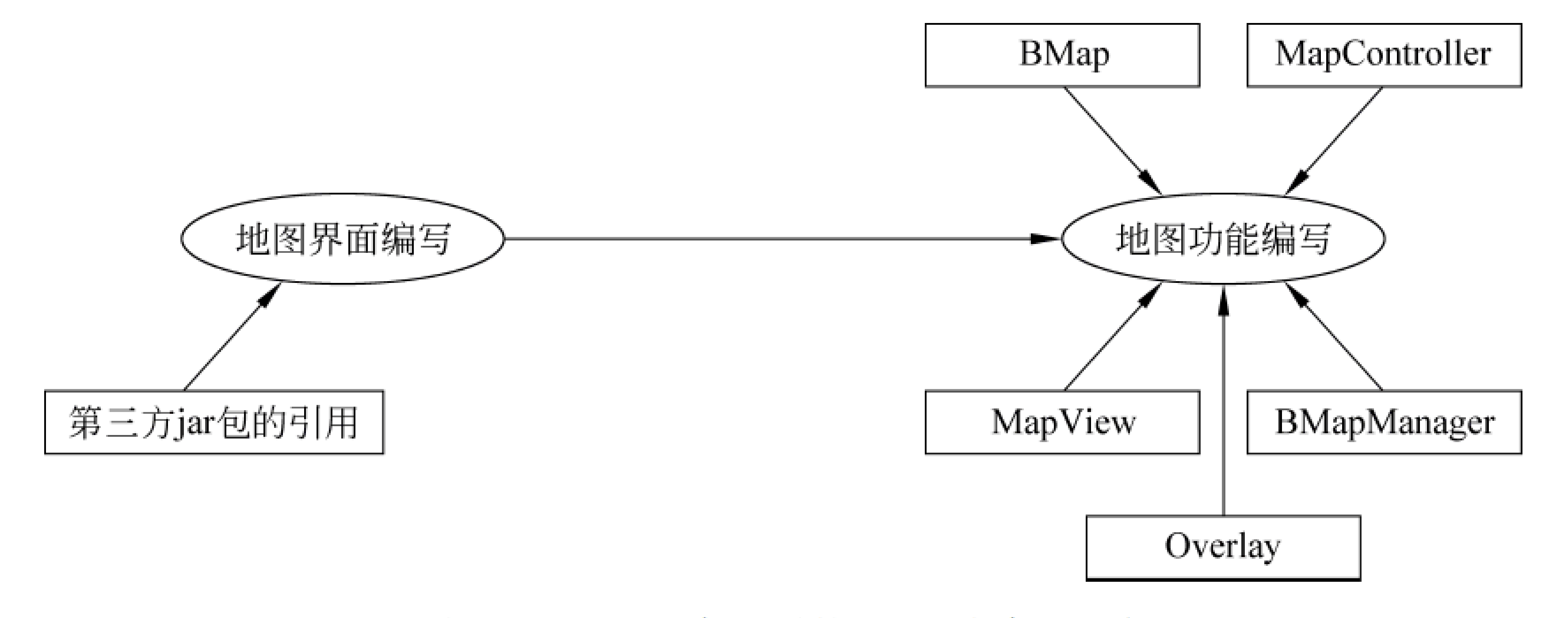


图 9-1 基于百度地图的 GPS 设计及开发

地图界面的编写需要正确的引入第三方 SDK 的 jar 文件,之后在 xml 中声明该 jar 包中的控件名,此处需注意的是在使用第三方或者自定义的控件时需要声明控件的全名(包括包名在内)。

地图功能的编写需要先初始化地图的控制类 MapManager,之后验证密钥。只有当密钥验证通过后才能使用百度地图的资源。然后初始化 MapView 类和 MapController 类,前者用来显示地图信息,后者用来控制前者的显示。OverLay 类用来在地图上设置一系列的覆盖物。

9.2 项目界面设计

9.2.1 知识准备

对于百度地图的展示和操作需要设计到用户隐私等,必须添加用户权限。

```
<uses-permission
android:name="android.permission.INTERNET"/>
<uses-permissionandroid:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permissionandroid:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permissionandroid:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permissionandroid:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permissionandroid:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permissionandroid:name="android.permission.READ_PHONE_STATE"/>
```

并且在定位时需要添加 service,下文会详细解释。

9.2.2 项目界面相关代码设计

(1) 地图视图界面设计

要想将百度地图在手机界面上显示出来,需要用到控件 MapView,即使用规定的名称 com. baidu. mapapi. map. MapView。

```
<RelativeLayoutxmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".ExampleDemo">

<com.baidu.mapapi.map.MapView
android:id="@+id/mv_information"
android:layout_width="wrap_content"
android:layout_height="wrap_content"/>

</RelativeLayout>
```

效果如图 9-2 所示。

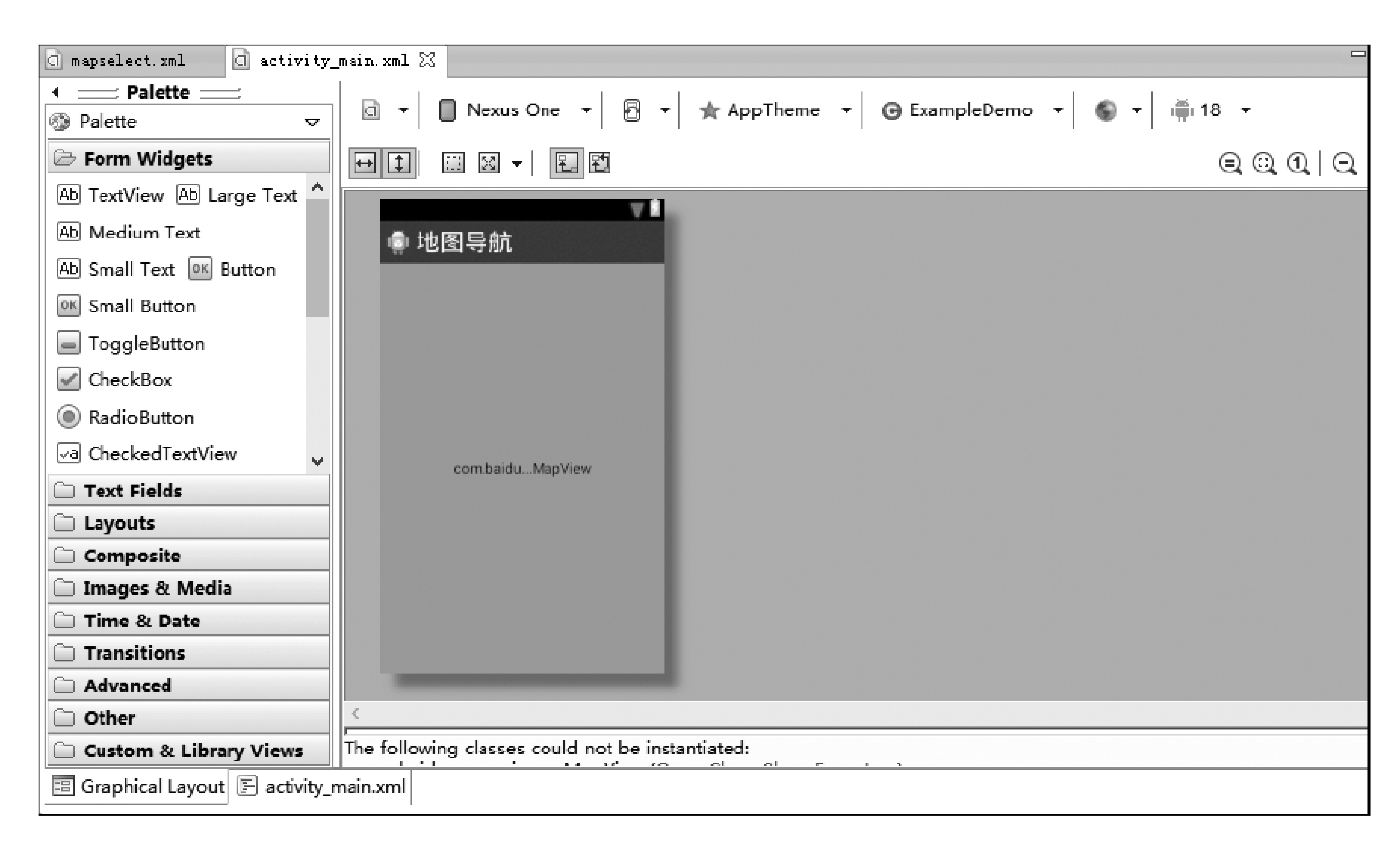


图 9-2　地图视图界面设计

(2) 子视图界面设计

子视图是用于在覆盖物上显示其详细信息的,需要控制视图和其中控件的大小。

```
<?xmlversion="1.0"encoding="utf-8"?>
<LinearLayoutxmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="horizontal"
android:paddingBottom="0dp">

<LinearLayout
android:id="@+id/ll_pop_info"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:orientation="vertical"
android:layout_weight="1"
android:background="#3f3f3f"
android:gravity="center_horizontal">

<TextView
android:id="@+id/tv_pop_title"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:ellipsize="end"
android:text="标题"
android:textColor="@android:color/white"
android:textSize="20sp"/>
```

```
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="我要去这里"/>
</LinearLayout>
</LinearLayout>
```

效果如图 9-3 所示。

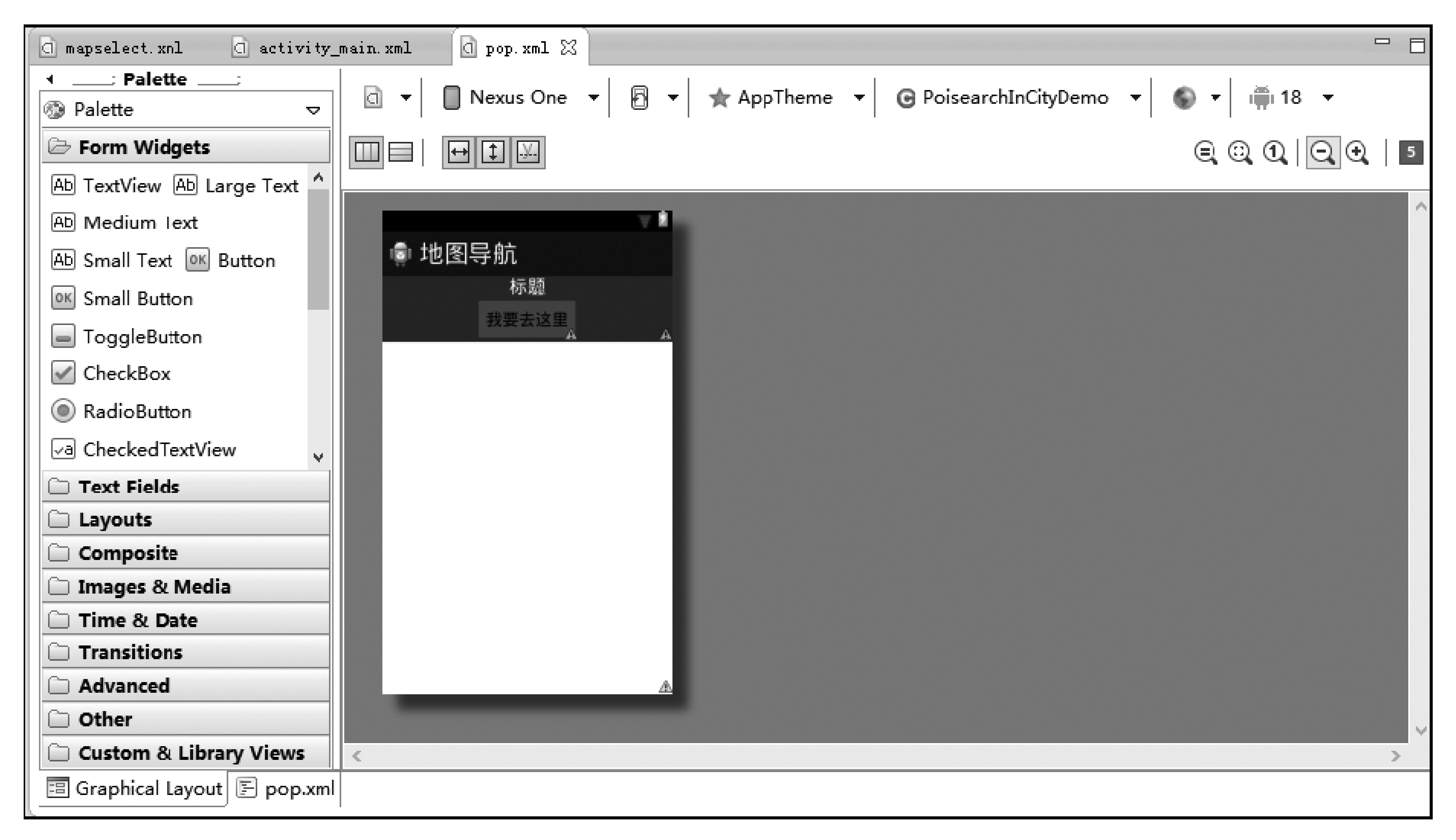


图 9-3 子视图界面设计

(3) 主界面设计

主界面为 ListView,单击不同的 item 跳转到对应的地图操作界面。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<ListView
android:id="@+id/iv_mapselect"
android:layout_width="fill_parent"
android:layout_height="wrap_content"></ListView>

</LinearLayout>
```

效果如图 9-4 所示。

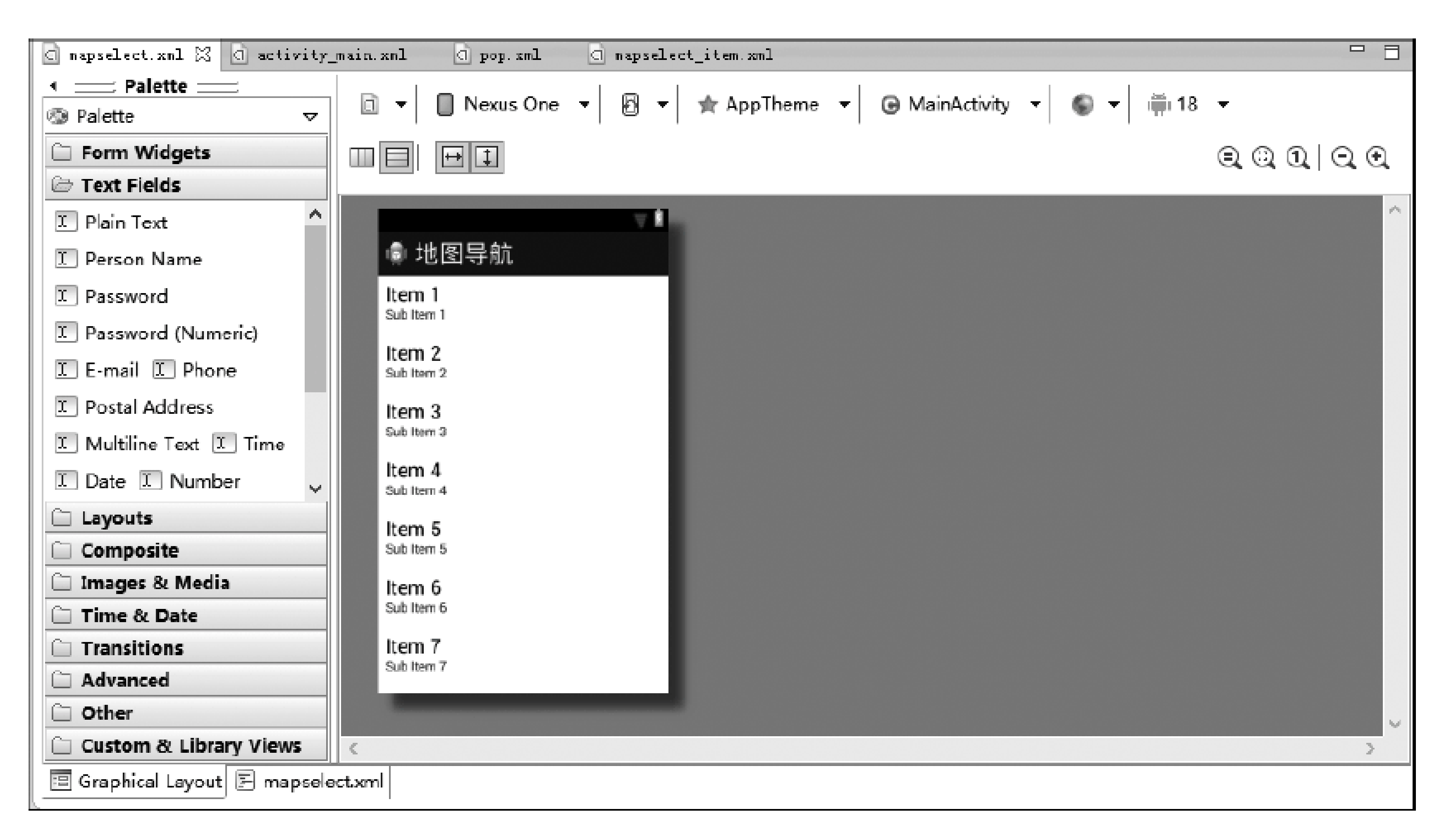


图 9-4 主界面设计

9.3 项目功能的实现

9.3.1 知识准备

SDK 下载：http://lbsyun.baidu.com/sdk/download。

KEY 申请：http://lbsyun.baidu.com/apiconsole/key(需要登录账号)。

申请 KEY 的流程如下：

(1) 登录系统账号。

(2) 进入我的应用并创建应用。

(3) 选择应用类型为 Android SDK 并配置安全码(eclipse 数字签名和包名)。

(4) 复制得到的 KEY 并加入清单文件。

流程如图 9-5～图 9-9 所示。



图 9-5 登录系统账号



图 9-6　查看应用界面



图 9-7　创建应用

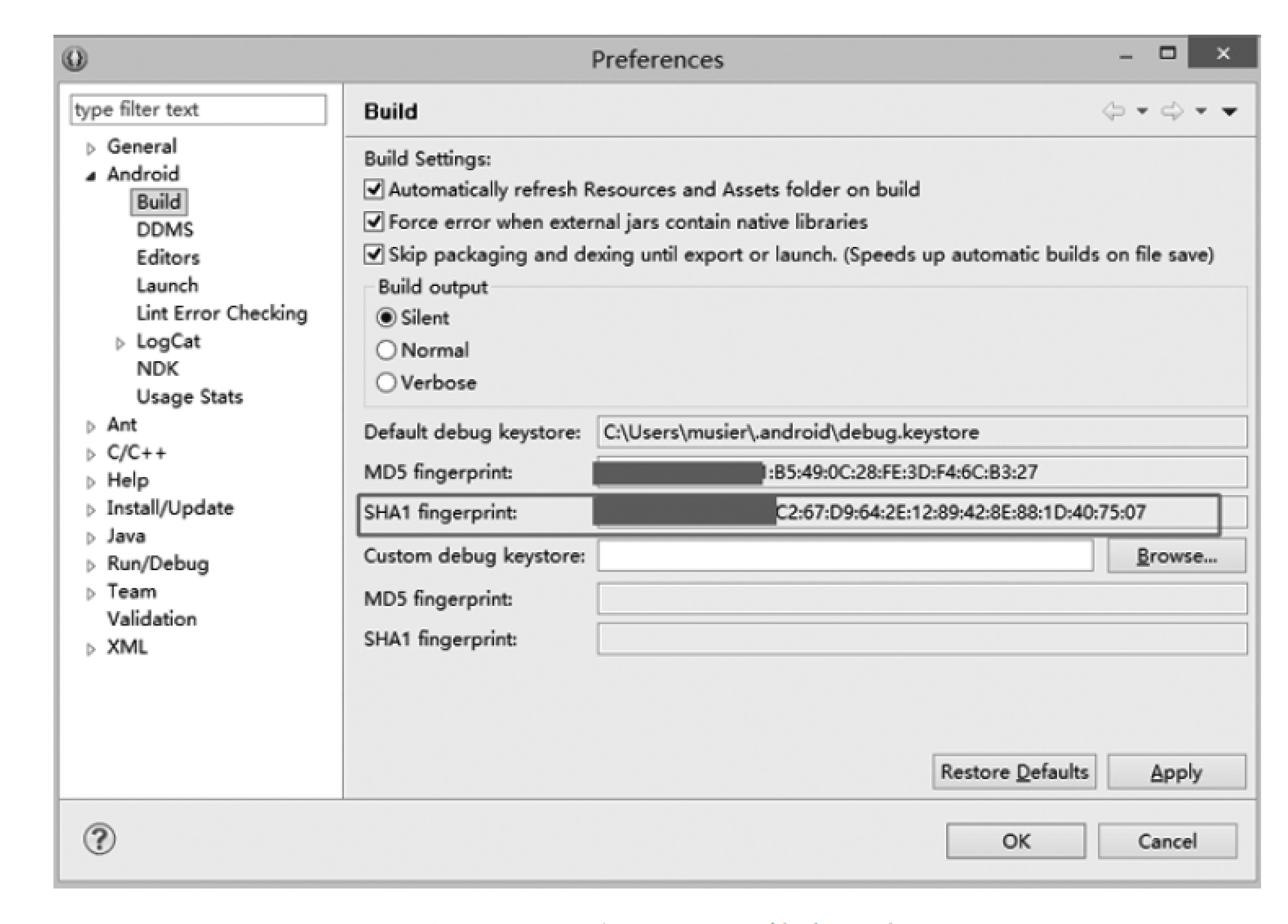


图 9-8　查看 Android 数字签名



图 9-9 得到 KEY

9.3.2 项目功能相关代码设计

（1）地图基本数据逻辑代码

① 地图三大核心类：BMapManager、MapView 和 MapController，依次分别为管理、视图和控制。

```
protected BMapManager manager;          //地图引擎管理
protected MapView mapView;              //地图的视图控件
protected MapController controller;     //控制地图：平移缩放旋转
```

② 在获取地图数据前必须要对申请的 KEY 进行验证，否则就会导致程序崩溃，此时需要用到 BMapManager 类。在 init()方法中设置监听事件判断验证是否通过和当前网路状况，其中的 PromptManager 类为工具类。

```
privatevoid checkKey() {
      manager= new BMapManager(getApplicationContext());
      //验证 key
      manager.init(ConstantValue.KEY, new MKGeneralListener() {

            @Override
            publicvoid onGetPermissionState(int iError) {
                  //授权验证
                  if(iError==MKEvent.ERROR_PERMISSION_DENIED){
                     PromptManager.showToastTest(BaseActivity.this,"授权验证失败");
                  }
            }

              @Override
            publicvoid onGetNetworkState(int iError) {
                  //没有网络
                  if(iError==MKEvent.ERROR_NETWORK_CONNECT){
                        PromptManager.showNoNetWork(BaseActivity.this);
                  }
            }
      });
}
```

③ 设置地图数据。主要为在界面上显示按钮、设置地图的默认视距和地图默认中心点等。需要注意的是，MapController 对象需要在 MapView 对象存在的情况下才可以得到。

```
privatevoid initView() {
    //TODO Auto-generated method stub
    mapView= (MapView) findViewById(R.id.mv_information);
    //显示内置放大和缩小的按钮
    mapView.setBuiltInZoomControls(true);

    //控制地图的缩放
    controller=mapView.getController();//必须先有 mapview 再有 controller
    controller.setZoom(12);
    controller.enableClick(true);
    controller.setCenter(point);//设置地图默认中心点
}
```

④ 为了在打开地图时可以及时地将数据展现在视图上而不至于出现混乱，需要将地图视图的生命周期与 activity 进行绑定。

```
@Override
    protectedvoid onResume() {

        mapView.onResume();
        super.onResume();
    }

    @Override
    protectedvoid onPause() {

        mapView.onPause();
        super.onPause();
    }

    @Override
    protectedvoid onDestroy() {

        mapView.destroy();
        super.onDestroy();
    }
```

⑤ 以下为 MKSearchListener 接口的实现，这样就可以直接继承该类并复写使用接口的方法而不至于每次在实现接口时同时实现所有的方法，从而提高效率。

```
protectedclass MySearchListenerAdapter implements MKSearchListener{

    @Override
    publicvoid onGetAddrResult(MKAddrInfo result, int iError) {
        //TODO Auto-generated method stub
```

```
    }

    @Override
    publicvoid onGetBusDetailResult(MKBusLineResult result, int iError) {
        //TODO Auto-generated method stub

    }

    @Override
    publicvoid onGetDrivingRouteResult(MKDrivingRouteResult result,
            int iError) {
        //TODO Auto-generated method stub

    }

    @Override
    publicvoid onGetPoiDetailSearchResult(int type, int iError) {
        //TODO Auto-generated method stub

    }

    @Override
    publicvoid onGetPoiResult(MKPoiResult result, int type, int iError) {
        //TODO Auto-generated method stub

    }

    @Override
    publicvoid onGetSuggestionResult(MKSuggestionResult result, int iError) {
        //TODO Auto-generated method stub

    }

    @Override
    publicvoid onGetTransitRouteResult(MKTransitRouteResult result,
            int iError) {
        //TODO Auto-generated method stub

    }

    @Override
    publicvoid onGetWalkingRouteResult(MKWalkingRouteResult result,
            int iError) {
        //TODO Auto-generated method stub

    }
}
```

效果如图 9-10 所示。



图 9-10　基本地图

(2) 地图图层逻辑代码

① 设置交通图的显示。

```
mapView.setTraffic(false);
```

② 设置卫星图的显示。

```
mapView.setSatellite(false);
```

③ 设置键盘单击事件监听切换不同的图层。

```
@Override
    publicboolean onKeyDown(int keyCode, KeyEvent event) {
        //TODO Auto-generated method stub
        switch (keyCode) {
        case KeyEvent.KEYCODE_1:
            //底图
            mapView.setTraffic(false);
            mapView.setSatellite(false);
            break;

        case KeyEvent.KEYCODE_2:
            //交通
            mapView.setTraffic(true);
            mapView.setSatellite(false);
            break;

        case KeyEvent.KEYCODE_3:
            //卫星
            mapView.setSatellite(true);
            mapView.setTraffic(false);
            break;
```

```
        }
        returnsuper.onKeyDown(keyCode, event);
    }
```

效果如图 9-11 和图 9-12 所示。



图 9-11　实时交通图



图 9-12　卫星图

(3）几何图形覆盖物逻辑代码

该类继承自基本地图视图类，这样就可以省去获取地图数据的代码，提高代码编写效率。

绘制覆盖物，一般对地图视图的操作分为四步：① 定义操作；②设置数据；③将操作对象加入视图中；④刷新视图。

```
privatevoid draw() {
    //定义覆盖物
    //设置覆盖物数据
    //获取 mapview 中存放覆盖物的集合,添加覆盖物
    //手动刷新界面

    GraphicsOverlay overlay= new GraphicsOverlay(mapView);
    setData(overlay);//重点:数据设置
    mapView.getOverlays().add(overlay);
    mapView.refresh();

}
```

为覆盖物对象设置数据。Symbol 为样式类定义操作对象的样式。

```
privatevoid setData(GraphicsOverlay overlay) {
    //定义几何图形:圆心+半径
    Geometry geometry= new Geometry();
    geometry.setCircle(point, 100);
    //定义几何图形的样式:颜色+线条等
    Symbol symbol= new Symbol();
    Symbol.Color color= symbol.new Color();
    color.red= 255;
    color.green= 0;
    color.blue= 0;
    color.alpha= 100;
    symbol.setSurface(color, 1, 0);

    //几何图形元素数据
    Graphic graphic= new Graphic(geometry, symbol);
    //设置数据
    overlay.setData(graphic);
}
```

效果如图 9-13 所示。



图 9-13 几何图形覆盖物

(4) 文字覆盖物逻辑代码

基本操作与几何图形覆盖物类似。

```
privatevoid draw() {
    TextOverlay overlay= new TextOverlay(mapView);
    setData(overlay);
    mapView.getOverlays().add(overlay);
    mapView.refresh();

}
```

对文字覆盖物的数据设置。

```
privatevoid setData(TextOverlay overlay) {
    TextItem item= new TextItem();
    item.align= TextItem.ALIGN_CENTER;            //对齐方式
    item.fontColor= getColor();
    item.fontSize= 20;
    item.pt= point;                               //坐标点
    item.text= "常信院";
    item.typeface= Typeface.DEFAULT_BOLD;         //粗体
    overlay.addText(item);

}

private Color getColor() {
    Symbol symbol= new Symbol();
    Symbol.Color color= symbol.new Color();
    color.red= 255;
    color.green= 0;
    color.blue= 0;
    color.alpha= 100;
    return color;
}
```

效果如图 9-14 所示。

(5) 多条目绘制逻辑代码

① 添加单击覆盖物时弹出的 pop,即子视图。因为未单击时不能确定坐标,所以暂时隐藏。

```
privatevoid initPop() {
    pop= View.inflate(this, R.layout.pop, null);
    titleTextView= (TextView) pop.findViewById(R.id.tv_pop_title);

    pop.setVisibility(View.INVISIBLE);

    //添加到 mapview 的容器里
    mapView.addView(pop);
}
```



图 9-14　文字覆盖物

② 设置 pop 显示时所展现的内容和显示的坐标点。

```
privatevoid draw() {
    //装载 OverlayItem 的集合
    ItemizedOverlay< OverlayItem> overlay= new ItemizedOverlay< OverlayItem> (this.getResources().
    getDrawable(R.drawable.icon_gcoding), mapView){

        //覆盖物的单击事件
        @Override
        protectedboolean onTap(int index) {
            OverlayItem item= this.getItem(index);
            String title= item.getTitle();
            titleTextView.setText(title);

            /**
             * 创建自定义布局参数,按地理坐标布局
             * 参 1:高度 参 2:宽度 参 3:坐标点 参 4:对齐方式
             * /
            MapView.LayoutParams params= new MapView.LayoutParams(
                    MapView.LayoutParams.WRAP_CONTENT,
                    MapView.LayoutParams.WRAP_CONTENT,
                    item.getPoint(),
                    MapView.LayoutParams.BOTTOM_CENTER
            );
            //更新 pop 的位置:将 params 与经纬度坐标建立关系
            mapView.updateViewLayout(pop, params);
            pop.setVisibility(View.VISIBLE);
            returnsuper.onTap(index);
        }
    };
    setData(overlay);
```

```
        mapView.getOverlays().add(overlay);
        mapView.refresh();
    }
```

③ 为覆盖物设置数据，因为 ItemizedOverlay 类实际为一个装载 OverlayItem 的集合，所以可以自行添加多个覆盖物。

```
    privatevoid setData(ItemizedOverlay<OverlayItem> overlay) {
        /**
         * 参 1:坐标 参 2:标题 参 3:介绍
         */
        OverlayItem item=new OverlayItem(point, "常信院", "计算机信息职业技术
         学院");
        overlay.addItem(item);
        item=new OverlayItem(new GeoPoint(latitude+1000,longitude), "向北",
        "增加纬度");
        overlay.addItem(item);
        item=new OverlayItem(new GeoPoint(latitude,longitude+1000), "向东",
        "增加经度");
        overlay.addItem(item);
        item=new OverlayItem(new GeoPoint(latitude-1000,longitude-1000),
        "向西南", "减少经纬度");
        overlay.addItem(item);

    }
```

效果如图 9-15 和图 9-16 所示。



图 9-15　多个覆盖物

(6) 附近兴趣点检索逻辑代码

① 之后的集合类涉及检索需要使用的 MKSearch 以及其监听 MKSearchListener。

```
    private MKSearch search;
```



图 9-16　覆盖物的单击事件

```
private MKSearchListener listener;
```

② 兴趣点检索。

```
privatevoid search() {
    search= new MKSearch();
    listener= new MySearchListenerAdapter(){
        @Override
        publicvoid onGetPoiResult(MKPoiResult result, int type, int iError) {
            //TODO 数据显示
            if(iError==0){
                if(result !=null){
                    PoiOverlay overlay= new PoiOverlay(PoiSearchNearByDemo.
                    this, mapView);
                    setData(overlay,result);
                    mapView.getOverlays().add(overlay);
                    mapView.refresh();
                }
            }else{
                PromptManager.showToastTest(PoiSearchNearByDemo.this, "未查询到结果");
            }

        }
    };
search.init(manager, listener);
```

③ 使用附近检索的方式检索加油站。

```
search.poiSearchNearBy("加油站", point, 5000);

}
```

④ 设置覆盖物数据。

```
protectedvoid setData(PoiOverlay overlay, MKPoiResult result) {
```

⑤ 得到全部检索到的兴趣点。

```
ArrayList<MKPoiInfo> datas= result.getAllPoi();
overlay.setData(datas);

}
```

效果如图 9-17 所示。



图 9-17 附近兴趣点检索

(7) 全城兴趣点检索逻辑代码

全城检索的逻辑步骤与附近检索基本相同,唯一的区别为全城检索自带分页。因为按城市检索出来的兴趣点数量太大而视图默认最多显示十条。

① 因为有分页所以需要清除旧数据。

```
PoiOverlay overlay= newPoiOverlay(PoisearchInCityDemo.this, mapView);
                    setData(overlay,result);
                    mapView.getOverlays().clear();
                              //有分页的情况下单击下一页后需要先清除上一次的检索记录
                    mapView.getOverlays().add(overlay);
                    mapView.refresh();
```

② 使用全城兴趣点检索加油站。

```
search.poiSearchInCity("常州", "加油站");
```

③ 为了能够清楚地看到分页情况,此处用子视图展示当前一共有多少条兴趣点和总页数。

```
protectedvoid setData(PoiOverlay overlay, MKPoiResult result) {
```

```
        ArrayList<MKPoiInfo> datas= result.getAllPoi();              //当前页的所有条目(默认 10条)
        overlay.setData(datas);

        String info= "当前页:"+ result.getPageIndex()+
                "/共"+ result.getNumPages()+
                "页,当前页条目数:"+ result.getCurrentNumPois()+
                "/总条目数:"+ result.getNumPois();

        PromptManager.showErrorDialog(PoisearchInCityDemo.this, info);
    }
```

④ 显示 current 值对应的页数,需要注意的是页码是从 0 开始的,所以在对翻页的最大、最小值进行限制时需要格外注意。

```
search.goToPoiPage(current);
```

效果如图 9-18 所示。



图 9-18　分页显示

（8）驾车或步行线路检索逻辑代码

① 设置监听事件并进行数据操作。

```
listener= new MySearchListenerAdapter(){
    @Override
    //此方法为驾车回调函数,如果是步行,需要使用 onGetWalkingRouteResult
    publicvoid onGetDrivingRouteResult(MKDrivingRouteResult result,
    int iError) {
        if(iError==0){
            if(result != null){
                //路线线条:驾车步行
                RouteOverlay overlay= new RouteOverlay(DrivingSearchDemo.
                this, mapView);
                setData(overlay,result);
```

```
                    mapView.getOverlays().clear();
                    mapView.getOverlays().add(overlay);
                    mapView.refresh();
                }

            }else{
                PromptManager.showToastTest(DrivingSearchDemo.this, "为检索到结果");}}};
```

② 将管理类和监听添加到 search 中。

```
search.init(manager, listener);
```

③ 设置起始点和终点,可以是名称或坐标点。

```
MKPlanNode start= new MKPlanNode();
start.pt= point;
MKPlanNode end= new MKPlanNode();
end.name= "春秋淹城";
//search.drivingSearch("常州", start, "常州", end);
```

④ 设置途经点,可以为多个,这样线路就会在起始点、途经点和终点之间形成。

```
ArrayList< MKWpNode> nodes= new ArrayList< MKWpNode> ();
MKWpNode node= new MKWpNode();
node.city= "常州";
node.name= "中华恐龙园";
nodes.add(node);

//驾车策略必须在搜索之前设置
search.setDrivingPolicy(MKSearch.ECAR_FEE_FIRST);
```

⑤ 使用驾车路线检索。

```
search.drivingSearch("常州", start, "常州", end, nodes);
```

⑥ 使用步行路线检索。

```
search.walkingSearch("常州", start, "常州", end);
```

效果如图 9-19 和图 9-20 所示。

(9) 公交换乘检索逻辑代码

```
listener= new MySearchListenerAdapter(){
```

① 公交换乘监听回调函数。

```
@Override
publicvoid onGetTransitRouteResult(MKTransitRouteResult result,
    int iError) {
        if(iError= = 0){
            if(result != null){
                //公交换乘需要使用的覆盖物
                TransitOverlay overlay= new TransitOverlay
```



图 9-19　驾车路线检索



图 9-20　步行路线检索

```
                (TransitOverLayDemo.this, mapView);
                setData(overlay,result);
                mapView.getOverlays().add(overlay);
                mapView.refresh();
            }
        }else{
            PromptManager.showToastTest(TransitOverLayDemo.this, "为查询到
            结果");
        }
    }
};
```

② 乘车策略。

```
search.setTransitPolicy(MKSearch.EBUS_WALK_FIRST);
```

③ 公交换乘检索函数只可以在本城市内检索，而步行和驾车可以跨城市检索。

```
search.transitSearch("常州", start, end);
```

效果如图 9-21 所示。



图 9-21 公交换乘路线检索

（10）定位功能逻辑代码

① 定位功能属于高级操作，需要 locSDK_3.1.jar 包的支持，并且在清单文件的 Application 节点下加入 service 节点。

```
<service
    android:name="com.baidu.location.f"
    android:enabled="true"
    android:process=":remote">
</service>
```

② 在地图数据显示之前设置定位数据并开启定位。

```
@Override
    protectedvoid onResume() {
        location();
        client.start();
        super.onResume();
    }
```

③ 在视图不可见时关闭定位。

```
@Override
    protectedvoid onPause() {
        client.stop();
        super.onPause();
```

```
    }

//设置数据如:间隔多长时间发送获取位置的请求等
        LocationClientOption option= new LocationClientOption();
        option.setOpenGps(true);
        option.setAddrType("all");        //返回的定位结果包含地址信息
        option.setCoorType("bd0911");              //返回的定位结果是百度经纬度,默认值 gcj02
        option.setScanSpan(5000);         //设置发起定位请求的间隔时间
        option.disableCache(true);        //禁止启用缓存定位
        option.setPoiNumber(5);           //最多返回 poi 的个数
        option.setPoiDistance(1000);      //poi 的查询距离
        option.setPoiExtraInfo(true);     //是否需要 poi 的电话和地址等详细信息

        client.setLocOption(option);

        client.registerLocationListener(listener);
```

④ 定位的监听,开启后在定位关闭前周期性回调。

```
privateclass MyDBLocationListener implements BDLocationListener{

    @Override
    publicvoid onReceiveLocation(BDLocation location) {
        if(location==null){
            return ;
        }

        //定位覆盖物
        MyLocationOverlay overlay= new MyLocationOverlay(mapView);
        LocationData data= new LocationData();
        data.latitude= location.getLatitude();
        data.longitude= location.getLongitude();
        overlay.setData(data);

        mapView.getOverlays().add(overlay);
        mapView.refresh();

        //模拟器定位
        controller.animateTo(new GeoPoint((int)(data.latitude * 1E6), (int)(data.longitude * 1E6)));

    }

    @Override
    publicvoid onReceivePoi(BDLocation arg0) {
    }

}
```

效果如图 9-22 所示。



图 9-22 定位功能

9.4 系统运行与效果测试

系统在模拟器和真机上均可流畅显示,但在真机上会有标注物减少和找不到检索点的情况,所以导致部分线路检索无法正常显示,需在以后的学习中进一步完善代码。

9.5 本章小结

经过本章的学习,初步了解了基于百度地图 SDK 的开发,能够在自己的项目中显示百度地图并且使用其中的基本功能:定位、路径规划和兴趣点检索等。结合这些基础的功能,可以自己发挥想象,创造出其他新颖有趣的功能。地图开发还包括一些其他的功能,如导航、语言播报等,这里不再赘述。如果有兴趣可以去开放平台下载对应的 SDK,参照官网的实例代码自己集成。

9.6 项目实践

(1) 把做好的 APP 安装到实验用手机测试基本功能。
(2) 使用不同分辨率的模拟器测试效果。
(3) 使用不同版本的模拟器测试效果。
(4) 在真机和模拟器上分别测试,看看二者在界面显示和功能上的一些区别。

参 考 文 献

[1] 李刚. 疯狂 Android 讲义[M].2 版. 北京：电子工业出版社，2013.

[2] 刘超. 深入解析 Android 5.0 系统[M]. 北京：人民邮电出版社，2015.

[3] Bill Phillips，Brian Hardy. Android 编程权威指南[M]. 王明发，译. 北京：人民邮电出版社，2014.

[4] 塞萨. 打造高质量 Android 应用[M]. 杨云君，译. 北京：机械工业出版社，2014.

[5] 李俊. 深入解析 Android 5.0 系统[M]. 北京：人民邮电出版社，2015.

[6] 安卓巴士. http://www.apkbus.com/.

[7] 慕课网(IMOOC). http://www.imooc.com/.

[8] eoe Android 社区. http://www.eoeandroid.com/.

[9] 百度地图开放平台. http://lbsyun.baidu.com/.